(4)

1986

Annual Technical Report

July 1985 - June 1986

A Research Program in Computer Technology

ISI/SR-87-178

U S C
INFORMATION
SCIENCES
INSTITUTE

4676 Admiralty Way - Suite 1001
Marina del Rey, California 90292
213 822 1511

(4)

1986

Annual Technical Report

July 1985 - June 1986

A Research Program in Computer Technology

ISI/SR-87-178

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT This document is approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISI/SR-87-178	5. MONITORING ORGANIZATION REPORT NUMBER(S) ———
6a. NAME OF PERFORMING ORGANIZATION USC/Information Sciences Institute 6b OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION ———
6c. ADDRESS (City, State, and ZIP Code) 4676 Admiralty Way Marina del Rey, CA 90292-6695	7b. ADDRESS (City, State, and ZIP Code) ———
8a. NAME OF FUNDING / SPONSORING ORGANIZATION DARPA 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903 81 C 0335
8c. ADDRESS (City, State, and ZIP Code) Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. ——— / PROJECT NO. ——— / TASK NO. ——— / WORK UNIT ACCESSION NO. ———

11. TITLE (Include Security Classification)
1986 Annual Technical Report:
A Research Program in Computer Technology (Unclassified)

12. PERSONAL AUTHOR(S) ISI Research Staff

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Research Report	FROM July 85 TO June 86	1989, August	150

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
09	02		(over)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the research performed by USC/Information Sciences Institute from July 1, 1985, to June 30, 1986, for the Defense Advanced Research Projects Agency. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Victor Brown Sheila Coyazo	22b. TELEPHONE (Include Area Code) 213/822-1511 / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

18. SUBJECT TERMS (continued)

1. artificial intelligence, automated implementation, Common LISP, expert systems, formal specification
2. domain model, domain principles, expert systems, integration knowledge, LISP, NIKL paraphraser, optimization knowledge, program writer, XPLAIN system
3. artificial intelligence, automatic programming, consistency maintenance, formal specification, programming environments, rapid prototyping, rule-based programming, software evolution
4. command graphics, computer graphics, high-level graphics language, network-based graphics, online map display
5. computer communication, electronic mail, internetwork protocols, protocol design
6. computer communication, multimedia conferencing, packet video, packet voice, protocols, video conferencing
7. computer communication, protocols, protocol design, supercomputers, Wideband Satellite network
8. CMOS/Bulk fabrication, MOSIS, printed circuit board fabrication, scalable design rules, VLSI
9. design rules, device fabrication, device testing, integrated circuits, MOSIS, silicon compilation, VLSI design, wafer testing
10. integrated circuit design, VLSI test equipment
11. classification-based reasoning, KL-ONE, KL-TWO, knowledge representation, NIKL
12. Janus, knowledge representation, natural language generation, Penman, text generation
13. computer mail, electronic mail, Intermail, mail forwarding, mail system interconnection
14. ARPANET, computer support, hardware, network services, software
15. ARPANET, computer communication, hardware maintenance, remote maintenance, remote monitoring
16. computer workstations, distributed processing, survivable networks
17. distributed processing, local networks, personal computers, workstation environment
18. computer acquisition, Strategic Computing
19. C3, computer communication, packet radio, survivable networks

ISI/SR-87-178

University of Southern California

1986

ANNUAL TECHNICAL REPORT

July 1985 – June 1986

A Research Program in Computer Technology

Principal Investigator and Executive Director:
Keith W. Uncapher

Prepared for the Defense Advanced Research Projects Agency
Effective date of contract 1 July 1981
Contract expiration date 31 May 1987
Contract # MDA 903 81 C 0335
ARPA Order 4242

INFORMATION SCIENCES INSTITUTE

213/822-1511

4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335 Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

CONTENTS

SUMMARY

This report summarizes the research performed by USC/Information Sciences Institute from July 1, 1985, to June 30, 1986, for the Defense Advanced Research Projects Agency. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

The ISI program consists of 19 research areas:

Common LISP Framework: producing an exportable version of the FSD testbed, which incorporates the well-understood portions of our research in FSD into a new automated software development paradigm, and distributing it to outside users; *Explainable Expert Systems*: creating a framework for building expert systems that enhances an expert system's explanatory capablities (allowing it to explain and justify its reasoning), and eases modification and maintenance of an expert system; *Formalized Software Development*: studying a new automated software development paradigm in which a formal operation specification, used as a rapid prototype, is evolved to have the intended functionality, and is mechanically transformed (with human guidance) into an efficient implementation; *Advanced VLSI*: providing access to 1.2 micron CMOS/Bulk fabrication, printed circuit board fabrication, and centralized functional testing of custom-designed parts prior to their assembly into packages and boards; *VLSI*: providing a low-cost, fast turnaround LSI/VLSI device fabrication service to support a geographically distributed VLSI research community with no direct access to VLSI fabrication but with access to a computer communication network, and conducting research on the VLSI design problem; *KITSER/VLSI Design Service*: developing cost-effective test devices for the DARPA VLSI design community, and transferring the technology to commercial companies willing to market and support the testers; *Command Graphics*: developing a device-independent graphics system and graphics-oriented command and control applications programs; *Internet Concepts*: exploring aspects of protocols for the interconnection of computer communication networks, specifically the design and implementation of an internetwork computer message system and the design of internetwork host and gateway protocols; *Multimedia Conferencing*: designing and developing an experimental multimedia real-time conferencing system based on packet-switched network technology, with the goal of enabling users to communicate interactively via workstations and the Internet in a combination of text, bitmap images, and voice media; *Supercomputer Workstation Communication*: providing protocols and prototype programs for effective use of high-capacity communication systems such as the Wideband Satellite network for applications involving remote access to supercomputers from powerful workstations; *Empirically Valid Knowledge Representation*: developing a language for programming intelligent applications, which will emphasize the construction of declarative models of application domains, with the aim of creating sharable and reusable knowledge bases; *Text Generation for Strategic Computing*: developing new methods for autonomous creation of text by machine, with the focus on fluent, easily controlled sentence and paragraph production, faithful representation of information from AI knowledge bases, and use of generated English in ongoing human-computer interaction; *Commercial Mail*: developing and operating a service to support the exchange of electronic mail between the Internet and various commercial mail suppliers; *Computer Research Support*: operating reliable computing facilities and continuing the development of advanced

support equipment: *DARPA Headquarters Support Center*: establishing a fully operational computer facility and a large conference room at the DARPA-ISTO offices in Washington, D.C., and providing remote monitoring and maintenance of the equipment; *Exportable Workstation Systems*: developing a remote testbed environment of advanced workstations and servers; *New Computing Environment*: exploring, determining, and implementing the next generation of computers and computing facilities for the ISI research environment; *Strategic Computing - Development Systems*: providing development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants; *Strategic C3 System Experiment Support*: participating in a Strategic Command, Control, and Communication systems experiment demonstrating and evaluating the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-based techniques).

1. COMMON LISP FRAMEWORK

Research Staff:
Robert M. Balzer
David S. Wile
Dennis Allard
Richard Berman
Ben Broder
Chloe Holg
Brent Miller
Donald Voreck
William Vrotney

Support Staff:
Audree Beal
Carol Sato
Jeanine Yamazaki

1.1 PROBLEM BEING SOLVED

The success of many components of the Strategic Computing program relies heavily on knowledge-based systems, systems in which some aspect of human knowledge about a problem area has been captured in rules for reasoning about and solving the problem. A number of commercial firms that specialize in providing such "expert system" tools have arisen, and many practical systems with military significance have been developed: for example, systems for equipment diagnosis, signal interpretation, and battle management. There are two approaches to improving services that are developed using expert system technologies. The first attempts to improve the language that expresses the knowledge used in the system and simultaneously to develop automatic tools to improve the performance and understandability of such systems.[1] The second, complimentary approach suggests substitution of task-specific, hand-tailored software for some of the knowledge sources in a system, where some problem feature makes their encoding as "rules" inappropriate or inefficient.

Although both approaches are important, this latter problem is especially important to ensuring the success of the SC program, wherein specialized equipment may have to be modelled with more fidelity than in more mundane expert system tasks. The Common LISP Framework project originally arose to support exactly this military problem: to facilitate SC contractors' development of problem-specific software in an expert system context. Almost every currently available knowledge-based system is programmed in some dialect of LISP. Hence, the SC program has procured LISP-based workstations for development of expert systems to effect the three primary military tasks it intends to accomplish. In order to maintain independence from the particular hardware environment on which the expert systems are developed, a standard dialect, called Common LISP, has been designed and will be required to run on each of these

[1] Such an approach is being pursued by the Explainable Expert Systems project at ISI.

supporting machines. This language will be used to capture the task-specific expertise used in conjunction with more general rules in a knowledge base. Although the use of the Common LISP language unifies expression of the knowledge in such systems, the diversity of hardware support environments makes the assumptions underlying the frameworks of such systems highly important. Code cannot be moved between hardware configurations unless it relies on the same underlying facilities, as well as the same language. The Common LISP Framework project provides a uniform framework across many vendors' hardware, in which SC contractors can develop and evolve their systems.

1.2 GOALS AND APPROACH

1.2.1 Goal: State-of-the-Art Framework for Common LISP Program Development

The principal goal of the Common LISP Framework project is to support Strategic Computing (SC) contractors with a comprehensive, state-of-the-art programming framework for the development and evolution of Common LISP programs. This framework provides the programmer with a fileless environment of persistent objects. Their relationships with other objects are manipulated via a set of generic operations and are used to associatively retrieve the objects. The Common LISP Framework system (the CLF) maintains the consistency of these objects and initiates automated processing for the programmer via a set of rules.

For several years ISI's Software Sciences Division has been developing the technology to support a new software lifecycle paradigm. This paradigm calls for the mechanical, but human-guided, derivation of software implementations from *formal specifications* of the desired software behavior. It relies on altering a system's specification and rederiving its implementation as the standard technology for software maintenance. Practical use of this paradigm requires extending active machine involvement into the earliest stages of the development process. The Formalized System Development project was established at ISI to study the feasibility of this approach on realistic-sized programs. It has developed a testbed system in which these ideas of specification and automated implementation may be studied through hands-on experimentation.

The ultimate goal of the Common LISP Framework project is to transfer, as it matures, this advanced technology for program specification, implementation via transformation, and reimplementation via replay of previously formalized developments on changed specifications. With the incorporation of successive versions of the FSD testbed into the framework, SC contractors will obtain the benefits of the paradigm. It is important to emphasize that we foresee many phases in which spinoffs from the FSD testbed are incorporated into the CLF for technology transfer.

1.2.2 Approach: Technology Transfer from the FSD Testbed

. The idea for the Common LISP Framework project arose as DARPA recognized the importance of the ongoing research in the Formalized System Development project at ISI, especially its potential for use by its SC contractors. The Formalized System Development Testbed System--hereafter called the testbed--is being developed to support research in domain modelling, expert systems, programming methodology, and administrative service development. Its major strengths are as follows:

- It provides a uniform, persistent, fully associative database to access all information. This constitutes a finer grain size than a file system and is the basis for integrating mundane, daily activities (mail, text editing, calendars, etc.) with the programming service.
- Within the database, the user can provide monitoring programs to maintain the integrity of his data automatically, and can provide rules that react demonically to changes in the state of the database to automatically accomplish activity that presently requires manual decision making and action.
- A consistent, model-based framework for interaction with the database is provided to unify generic activities such as editing, viewing, scrolling, deleting, and focusing.
- Program construction is done via the cycle of specification, examination of behavior, and transformation to implementation, followed by respecification, reexamination of behavior, and retransformation using the previously recorded development history.

Three features distinguish the testbed from other programming environments:

1. It supports associative retrieval in a persistent object-base whose consistency is automatically maintained by rules.
2. It actively, automatically supports mundane programmer activity via demonic rule invocation based on object-base changes.
3. It supports program evolution and maintenance via recording and structuring a "development history" of the programming process.

The system is being used to develop itself, forcing it to mature in realistic directions. Although research problems continue to abound in the development of this system, this maturity made it ripe for technology transfer through this effort.

The FSD testbed is designed to support all aspects of computing, from the mundane activities of mail handling and document preparation, through programming, to advanced software methodology experimentation. The major role of the Common LISP Framework project was initially to extract that portion of the environment that will be universally useful to the SC community. Identifying that subset and making it robust enough for real use constituted the major problem for the Common LISP Framework

project in its first year. Although the project only began late in 1984, a preliminary version of the CLF was in beta test by the end of 1985 and delivered to other SC contractors in early 1986.

1.3 SCIENTIFIC PROGRESS

From the outset, the Common LISP Framework was intended to go through three major phases. The functionality of the initial CLF can be best characterized as flexible management of program objects and tools. During the second phase, the tools used to provide the initial system were made more reliable and robust, and released for use in user programs. The third phase will begin to incorporate the FSD technology for program specification, implementation via transformation, and reimplementation via replay of previously formalized developments on changed specifications.

1.3.1 Salient Features of the CLF

The Common LISP Framework comprises the following major components:

- AI Operating System (CLF Kernel)
- Program Management Service
- Program Development Service

The AI operating system kernel of the CLF provides the programmer with a fileless environment of persistent objects. Their relationships with other objects are manipulated via a set of generic operations and are used to associatively retrieve the objects. The CLF maintains the consistency of these objects and initiates automated processing for the programmer via a set of rules.

A unique feature of the CLF is that all objects manipulated by programmers are represented in this persistent object-base--specifically structured objects, like modules with components, as well as primitive program structures such as function, variable, and structure declarations. Also, relationships and objects elsewhere represented in a very ad hoc, diversified fashion, such as flow-analysis relationships, versions, time stamps, developers, and users, are all represented uniformly in the object-base.

The kernel provides generic facilities for manipulating these programming objects as objects. In addition, special facilities have been introduced into the framework to provide programming assistance in the following areas:

- Module creation
- Component addition
- Importing existing files
- Component modification and editing

- Code installation

Each of these facilities requires user intervention or initiation.

Especially important are the facilities for managing the consistency of program information, built using the rule-based kernel. These facilities perform the following activities:

- Automatically compile functions that are "installed" by the user
- Allow multiple-buffer editing of the same object while maintaining correct views in each buffer
- Automatically (re)analyze functions when they are installed
- Automatically maintain program object ordering by load-order and view-order

Each of these activities is managed differently in existing programming environments. Some are managed by the user only--the system provides no help for such consistency maintenance.

The CLF's Program Development Service provides automated maintenance documentation by maintaining an annotated development history. The user is responsible for providing development step annotations briefly characterizing his activity when he changes the program. The programmer may explicitly indicate substructure in his development, whereupon the system maintains his stated goal structure.

Of considerable importance is the ability of a user to indicate his plans for future programming activity, through creation of development steps called "pending steps." The user can subsequently--perhaps at a much later time--handle these development steps: the system automatically incorporates them as new steps in the existing structured history.

Not only is the development service able to recall the history of the programming activity itself, but it is linked into the persistent object-base management activity in such a way that information associated with the changes may be used for maintenance documentation and release and for version management. Thus, since the generic object-base facility is used to store the development history *itself*, one can find all the development steps affecting a particular object, as well as all the objects affected by a particular development step.

This link into the incremental saving mechanism allows the development service to provide automated distribution of program objects to users of the systems of which they are components, as well as to aid in tracking the installation of sets of changed program

objects when the affecting development steps are accepted. The documentation used to describe development steps is used to document the distributed changes to users of the system. This is a particularly interesting side-effect of our efforts to record as much as possible of the programming process in the machine, where the information can be analyzed and the user aided based on this analysis.

1.3.2 Accomplishments

The major accomplishments of the Common LISP Framework project in the reporting period are detailed below.

An initial version of the CLF was released for beta test to two sites. This version of the system included the following:

- The relational objectbase of program objects (modules, functions, record declarations, etc.) and analysis predicates (flow relationships, call relationships, variable usage patterns) via which all tools communicate.
- The testbed facilities for creation, modification, destruction, retention, organization, and retrieval of these objects.
- The interactive standardized user interface to these facilities using a menu/window system.
- Tools that interface to these objects, including the ZMACS editor and a batch-invoked tool for program- and data-flow analysis.
- The "DEVELOP" mechanism to record a history of change within the FSD environment.
- A facility for converting existing programs for use in the framework.
- A preliminary facility for managing system releases.

This version of the system was subsequently released, along with documentation consisting of a brief manual and an extensive on-line, self-guided tutorial introduction to normal use of the system.

Although the initial CLF provided flexible data management of all program objects and tools, it suffered from two major problems:

1. It did not run efficiently enough for large program development.
2. Users were not provided access to the underlying object manipulation mechanism necessary for them to extend and customize CLF.

Both problems arose from limitiations with the released version of the underlying objectbase manager. In response to these deficiencies, a new version of the objectbase manager was developed in FSD and then migrated to CLF.

The remainder of the year was spent improving the facilities provided in the initial

version of the system and in giving programmer access to some of the foundational facilities used by the CLF:

- More user access to functionality underlying the delivered system was released in the form of a programmatic interface to the objectbase.
- Incremental flow analysis and compilation facilities were added.
- We began to keep track of activity histories, allowing undo, redo, etc.

In addition, two "unplanned" accomplishments are worth emphasis. Of particular significance is the "production mechanism" for versions of the CLF. As originally proposed, this mechanism was to be an "extraction" of functionality from the FSD environment. In fact, we reached our goal of converting CLF itself into Common LISP much earlier than we intended--largely to support in-house use of the environment on disparate machines. Thus, CLF actually forms the kernel for FSD; that is, FSD is built on top of CLF, rather than CLF being extracted from FSD.[2] This is actually quite important, since it means that CLF is being used in the testbed. Thus ISI researchers can act as the "beta test" of CLF releases.[3] This allows us both to speed up the release cycle and to improve the testing of CLF releases.

Another unscheduled activity that has proved beneficial to both ISI and DARPA has been the conversion of the CLF to two additional LISP workstations. Originally, the CLF was proposed only to work with Symbolics 3600s. DARPA's subsequent purchase of a number of different LISP workstations and ISI's receipt of a significant number of grant machines (from hardware manufacturers) made it both important and sensible to port CLF to other Common LISP workstations. The TI Explorer and the HP 9000 model 320 were chosen as the initial workstations to port CLF to. These porting efforts are well under way.

1.4 IMPACT

The Common LISP Framework project directly supports the Strategic Computing program in a very fundamental way. We emphasized earlier the extent to which the Common LISP Framework provides a uniform base for development of expert systems in the various programs funded by the Strategic Computing program and how the Common LISP Framework can enhance expert systems' capabilities by providing for the substitution of task-specific, hand-tailored software for some of the knowledge sources in a system, where some problem feature makes their encoding as "rules" inappropriate or inefficient.

---

[2]Naturally, some minimal renaming of symbols is required to provide the external view (as CLF rather than FSD).

[3]CLF is provided free to DARPA contractors. A graded price schedule is used for distribution to institutions and corporations.

In fact, the Common LISP Framework has a secondary benefit, which ultimately could be more important than its original purpose. The Common LISP Framework project provides an advanced, object-oriented programming environment for the development of Common LISP programs and a framework of conventions, structures, and models into which users can integrate their software. Such a powerful programming environment has never been available to anyone, let alone the military. Its availability to programmers should improve their output, and decrease the time and money needed for maintenance of programs written for military and civilian purposes. In addition, the framework will be used as a model environment for more compilation-oriented languages, like ADA, already proven to be of military benefit.

1.5 FUTURE WORK

CLF is basically a technology transfer project, adapting well-understood aspects of the FSD system to Common LISP programming support. The development cycle of functionality obeys the diagram in Figure 1-1. As each system feature matures in the

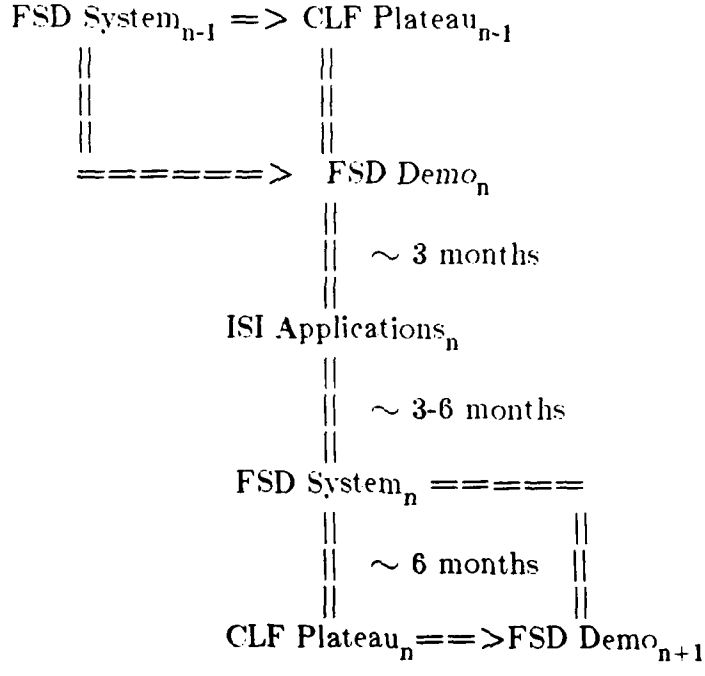


Figure 1-1: FSD-CLF plateau cycle

FSD system, it becomes incorporated into the CLF. That CLF facility then becomes the foundation for the next version of the FSD system. Generally, features undergo two distinct phases. The initial "demonstration" phase (FSD Demo$_n$) is incomplete with respect to some intended functionality, but demonstrates a portion of that intended functionality as a straw man--a rapid prototype. After about three months, the demonstrated features are robust enough to endure "alpha test" in ISI applications. They then become official FSD system features (FSD System$_n$) and are "beta tested" at

ISI for several months by a wider user community before being incorporated into the CLF (CLF $Plateau_n$). This plateau then becomes the base of the next FSD demonstration.

The next two plateaus planned for in the FSD system are the local annotation plateau and the independent specification language plateau. At the *local annotation* plateau, specifiers will be able to use a pure extension of Common LISP as their specification language. This extension will permit them to omit, selectively or totally, any specification of target language data structures to be used, and to specify logical tests and data retrievals over their data with full first-order logic, without regard to their procedural realization. The specification language will also provide for the expression of logical consistency conditions that must be maintained by the implementation, and for data-driven invocation of procedures (automation rules). Annotations added to these specifications will guide an automatic compiler in producing pure Common LISP code that implements them. Human implementors will select annotations with some knowledge-based assistance. This assistance will include, but will not necessarily be limited to, keeping track of the annotations that must still be added to make the specification compilable, providing menus of annotations that are applicable to selected parts of the specification, and prohibiting selections of incompatible annotations.

At the *independent specification language* plateau, the specifier will be using a notation that is no longer an extension of the target programming language, and that does not rely on the use of target language procedures for parts of the specification. This language will have a grammar-defined syntax. Functional and procedural abstractions will at least permit, and possibly require, typed inputs and outputs, to support greater analytical support at the specification level. The language will be one that has an extensible type set and multiple inheritance. Implementation will still be by annotations to the specification, but many annotations will have the form of transformations, making it possible to add new annotations without altering the compiler. The power of the specification notation will still be compromised sufficiently to permit treatment of the annotations as an unstructured collection of pieces of advice.

In the next reporting period, several "specification-based" facilities will be introduced to permit and effect the separation of specification of functionality from its implementation. The primary mechanism is a compiler "annotation" mechanism that instructs the compiler to choose particular representations for abstract data structures, and to optimize queries based on size and effort estimates for particular structures. To effectively use this capability, users must understand the effects of annotations without actually seeing the implemented structures. Hence, an "anotation analysis" mechanism will be introduced to provide efficiency estimates and measurements to guide the annotation process. Another aid to specification is the ability to manipulate them through their grammatical structure. Portions of the Popart system, our BNF-driven

program synthesis support mechanism, are currently in alpha test in the FSD testbed; they will be incorporated into the CLF to aid users in this area.

2. EXPLAINABLE EXPERT SYSTEMS

Research Staff:
William R. Swartout
Robert Neches
Steve Smoliar

Research Assistants:
Johanna Moore
Jody Paul

Support Staff:
Audree Beal

2.1 PROBLEM BEING SOLVED

The Explainable Expert Systems (EES) project has two goals for building expert systems: to extend and enhance the range of explanations that they offer, and to ease their maintenance and evolution. These goals are highly complementary because they place similar demands on the underlying architecture of the expert system.

Expert systems will play an important and expanding role in the near future. However, even the best expert system is worthless if it is not accepted by its intended users. A critical factor for user acceptance of expert systems is that they must be able to explain their reasoning. Recent surveys of potential users of expert systems (for example, [12]) have shown that, among the capabilities surveyed, *explanation* was regarded as an essential capability--more important than automatic knowledge acquisition, common-sense reasoning, ease of use, or even being error-free. An explanation facility should be able to assist the user in the following ways:

- give the user confidence in the system's results
- warn the user if the system is being applied inappropriately
- aid system developers in debugging/developing the system
- help to educate the user

Many expert systems provide no explanations at all. Those that do usually explain how they performed various tasks by paraphrasing the rules or methods they used into English [7, 8]. For this to be successful, those rules or methods must be written in a stylized fashion; and the techniques that the system employs have to be relatively close to those that users would be familiar with, since the system's paraphrases mirror the code directly. An advantage of this approach is that, since the explanations are provided by translation of the code itself, any changes to that code are immediately reflected in the explanations--so the system's documentation is always consistent with the system itself.

Unfortunately, the technique of paraphrasing the code is limited to describing *what* a system does or did. It cannot provide good explanations of *why* the system did what it did, that is, justifications of the system's actions. The problem is that the knowledge needed to support such explanations, the "rationale" behind the code, is not represented

in the code itself and hence is unavailable to the paraphrasing routines. This is an important limitation, because a critical factor in being a good consultant (human or computer) is the ability to justify one's recommendations so that clients will have confidence in them and follow them.

Let us illustrate some of these problems with an hypothetical example from the domain of digital circuit diagnosis.[1] Figure 2-1 shows a simple digital circuit consisting of two adders and three multipliers. Suppose that we constructed an expert system to diagnose this circuit (which we will call DIAGNOSER) and we used the translate-the-code approach to give it an explanatory capability.

In Figure 2-2 we see the sorts of limitations that the translate-the-code approach suffers from in providing justifications. Here, the system has asked the user to enter the signal at A1-OUT1; and the user, rather than entering an answer, has asked "why?" indicating that he wants to know why the question is being asked. The explanation shown could be produced by translating the expert system's program stack. While the explanation given does suggest that the information will be used to determine whether A1 is faulty or not, if the user wants a causal argument about why checking A1-OUT1 will indicate whether or not A1 is faulty, he's out of luck because that knowledge isn't represented in the system. The knowledge isn't represented because it doesn't have to be for the program to *perform* correctly. Just as one can follow a recipe and bake a cake without ever knowing why the flour or baking powder is there, so too can an expert system deliver impressive performance without any representation of the reasoning underlying its rules or methods. For DIAGNOSER to mimic expert behavior, all it needs to know is to ask for the value of the signal at A1-OUT1 and conclude that A1 is faulty if the value is incorrect. However, if we want to *justify* that behavior and explain why A1 is faulty if A1-OUT1 is not equal to 6, much more knowledge is needed: the system must know that A1-OUT1 is the output of A1 and that its expected value is 6. Further, the system must know that a component is faulty by definition if its actual output is not equal to its expected output and its actual inputs are equal to their expected values. This gap between the knowledge needed for performance and that needed for explanation is a recurring problem whenever one tries to retro-fit an explanation capability to an existing *performance* expert system.

The development of an expert system is an evolutionary process. We usually lack sufficient foresight to precisely specify, prior to a system's construction, what it should do. We form a better understanding of both the problem and the requirements for the system through experimentation with a prototype that we modify and extend until it performs adequately. Due to this experiential, incremental nature of development, it is

---

[1] For similar examples from actual working systems see [8, 7].

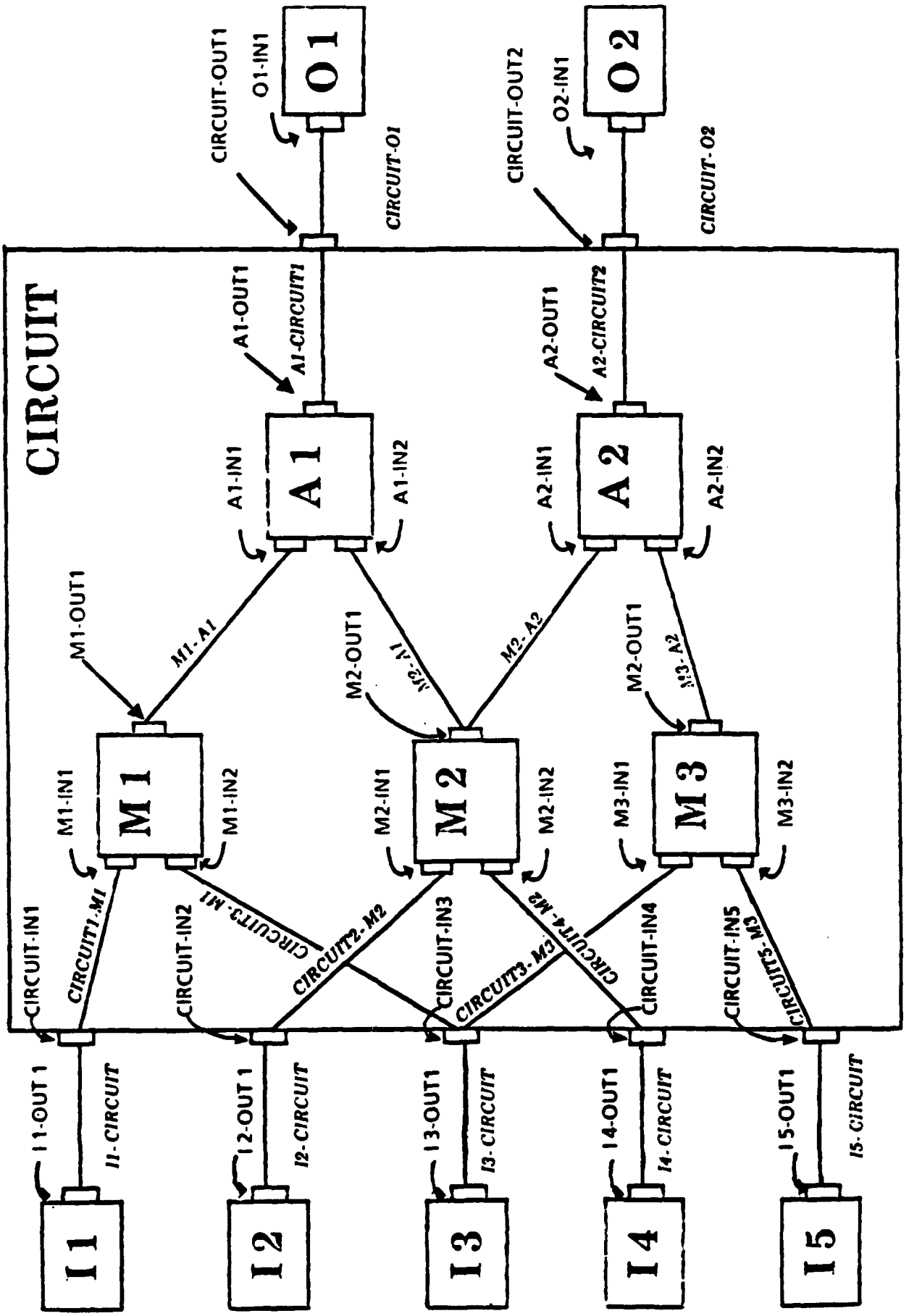


Figure 2-1: Simple digital circuit testbed

System: Please enter the value of the signal at A1-OUT1:

User: Why?

System: DIAGNOSER is trying to determine whether adder A1 is faulty.
If the value of the signal at A1-OUT1 is not equal to 6, then DIAGNOSER
will conclude that A1 is faulty.

Figure 2-2: A hypothetical example of limited explanations by paraphrasing the code

critical that an expert system shell support the addition, modification, and deletion of knowledge with minimal disturbance to the rest of the knowledge base; and that a developer be able to determine how a proposed change will affect the behavior of the system.

What is needed for evolvability? There seem to be two major factors. First, the expert system organization should be *modular* in the sense that it should be possible to change one part of the system independent of the rest of the system. Second, the expert system organization should be *explicit*, so that the effect of a change can be readily understood. A system is least evolvable if change to one part of the system affects other parts and does so through implicit mechanisms.

Advocates of rule-based expert system shells argue that this desired modularity arises inherently from the use of rules [2]. However, practical experience indicates that merely adopting a rule-based framework does not guarantee modularity and in some ways can impede it [1].

We feel that problems in both explanation and maintenance stem from related fundamental flaws in the underlying expert system architecture:

- **A weak knowledge representation with no separation of concerns.** Current frameworks do not support a modular representation of the knowledge in an expert system. The rules or methods incorporate many different kinds of knowledge, such as domain facts, heuristics for achieving goals, efficiency concerns, and so forth, that should be represented separately and explicitly but never are. As a result, the explanations such systems can provide are often opaque. This intertwining of knowledge also reduces a system's modularity and makes it more difficult to modify.
- **No record of operationalization of abstract concerns.** The process of compiling general domain facts and problem-solving knowledge into specific rules is performed, unrecorded, in the system builder's mind. The design decisions behind the system are not available; hence, machine-produced justifications of its behavior cannot be provided. This also makes the system more difficult to modify or evolve, because the system builder is forced to work at a low level of representation, mentally compiling high-level changes into low-level rules.

- **Limited explanation techniques.** Even when a framework does provide an explanation facility, the facility usually suffers from some serious limitations. In particular, such facilities are inflexible, and usually provide only one way of presenting an explanation. They are difficult to modify because the explanation techniques are currently hand-coded in LISP. Finally, they not responsive to the user's needs. If the user doesn't understand the explanation, there is no way for him to indicate to the system exactly what it is that he doesn't understand.

2.2 GOALS AND APPROACH

The EES project has created a framework for building expert systems that captures the knowledge necessary for providing better explanations and eases the evolution of an expert system. Based on the observation that the person who wrote an expert system can usually provide good justifications for its behavior, we have developed a new paradigm for expert system construction. In this paradigm, system builders and domain experts collaborate to represent, in a high-level specification language, the various kinds of knowledge that go into an expert system, such as knowledge that describes the domain, problem-solving knowledge, and terminology. An automatic programmer then uses that knowledge base to create an expert system (in LISP) from a high-level goal. As it creates the expert system, the automatic programmer records its design decisions in a *development history.* This development history provides the rationale behind the code that is needed to provide justifications and richer explanations of the expert system's actions. The EES project builds on the approach first adopted in the XPLAIN system [9] and extends it with a more advanced knowledge representation that allows us to represent additional kinds of knowledge in the high-level specification language, and a more powerful program writer that is capable of performing reformulations.

2.3 SCIENTIFIC PROGRESS

In developing the specification language for EES, we began by determining what kinds of explanations expert systems should offer. Using protocols and our own experience as expert system builders and users, we identified approximately a dozen different classes of useful explanations (see [10]). Based on that investigation, we identified different kinds of knowledge needed to support such explanations. We have provided representations for several of them, including domain-descriptive knowledge, problem-solving knowledge, and terminology [4, 5]. EES uses NIKL [3] as its underlying knowledge base. This provides one of the key advances of EES over XPLAIN: the explicit separation of terminology. We have identified this as a critical factor in enhancing the evolvability and explainability of expert systems [11].

Our specification language distinguishes different kinds of knowledge that embody expertise (see [5] for a discussion of how these distinctions were drawn):

- *Problem-solving knowledge* is "how-to" knowledge that includes *goals*, stating what is to be done, and *methods*, consisting of sequences of steps for achieving goals. It is procedural in nature and can thus support explanations based on the ability to trace the operations of its procedures.
- *Domain-descriptive knowledge* can be thought of as the "textbook rudiments" that are required before one can turn to solving problems. In an expert system for diagnosing a digital electronic circuit, this would include knowledge of the behavior of the different kinds of devices used and the ways they are connected. Explaining why a problem-solving method works must involve explaining how that method constitutes an implementation of such knowledge.
- *Terminology* is a lexicon of symbols used in the representation of both problem-solving knowledge and domain-descriptive knowledge. For example, the digital electronics domain would include terms like **system**, **input**, and **adder**.

The EES program writer is now operational. The program writer is the central component of the EES system. Using a high-level specification for an expert system, the program writer can produce a LISP implementation for the expert system. The program writer uses two passes. During the first pass, the basic structure of the expert system is designed, and that design is recorded in a machine-readable development history. The second pass of the system examines the recorded design and produces the actual LISP implementation. The first pass is substantially more complex and has been tested on two domains: a software tool called the *Program Enhancement Advisor* and a diagnostic system for telemetry systems. The second pass of the program writer has been tested on the second domain and has produced a demonstration expert system of approximately 10 pages of LISP code.

The program writer primarily creates the implementation by refining high-level goals into increasingly lower level goals until the level of system primitives is reached. The major advance of the EES program writer over XPLAIN's program writer [9] is that the EES program writer can reformulate a goal into other goals if no plan can be found for implementing the goal. The program writer makes use of several kinds of reformulations that seem to occur frequently in expert systems [5]. Providing for reformulation has had several benefits. First, it allows us to more accurately model the process of creating an expert system, because the problem of reformulating desired goals in light of available implementation technology is in fact a major part of expert system design. Second, by further relaxing the coupling between plans and goals, this approach should facilitate knowledge reuse across expert systems.

2.4 MILITARY IMPACT

The EES project is an important step in assuring that expert systems can be created and suitably maintained by the military without contractor assistance. EES provides leverage in three important ways:

1. EES's explanatory capabilities can be used by a maintainer to better understand an expert system's current capabilities and the effects of proposed changes.
2. Expert systems created using EES are inherently more modular than those created using conventional expert system shells. This makes it easier to change one part of the system independent of the rest (and opens up the possibility of knowledge reuse).
3. The EES approach will provide a permanent record of major design decisions by the designer and of subsequent decisions by others improving a particular expert system developed under EES. With the frequent turnover of military personnel, such a record should be invaluable.

2.5 FUTURE WORK

We will extend the EES framework in three areas:

1. **Specification.** As mentioned above, we have identified several kinds of explanations that expert systems should be able to provide. To provide some of these, it will be necessary to capture additional kinds of knowledge in the underlying specification language that EES provides for constructing expert systems. This will allow us to provide these explanations and, by increasing the modularity of the expert systems, make them more maintainable.
2. **Operationalization.** Because EES explicitly provides a specification and an implementation for an expert system, as well as a development history that connects the two, we believe we will be able to mimic human expert problem-solving behavior more closely than can be done with conventional expert systems through the use of "compiled-in assumptions," which we describe below.
3. **Explanation.** We want to provide a flexible explanation facility that can fully exploit the knowledge represented within the EES framework. Further, we want this explanation facility to be capable of reacting to user misunderstandings and presenting additional clarifying explanations.

The remainder of this section discusses our plans in greater detail.

2.5.1 Specification

The major improvement we want to make to EES's specification language is to augment it so that it represents the knowledge needed to answer questions such as the following:

1. What are the problems that may be solved (that is, what are the *goals* that the problem solver knows about achieving)?
2. What is the *intent* behind a goal (that is, what does it mean to solve a particular problem)?
3. How do the *actions* taken in solving a problem contribute to the ultimate solution (that is, how do the problem-solving methods work)?

Answering questions such as these requires the knowledge that underlies EES's plans, but because EES does not represent that underlying knowledge, these questions cannot be answered at present.

Our first experiment with EES involved the construction of a demonstration-sized expert system for the diagnosis of telemetry systems [11]. While this experiment demonstrated the feasibility of our approach and captured the knowledge needed to answer several kinds of questions that previously could not be answered [5], it was not possible to answer the question, "What does it *mean* to diagnose a system?" Problem-solving knowledge of *how* to diagnose could be retrieved, but it could not be related to domain-descriptive knowledge that characterized *what* a diagnosis was.

For example, the system had several methods for diagnosing a system, which we have hand paraphrased in Figure 2-3. These display what the system does in performing a diagnosis, but it takes considerable deductive effort on the part of the user to figure out what a diagnosis amounts to. We would like to have some explicit representation of the intent behind a diagnositic goal, such as: "To diagnose a decomposable system means to find a primitive subcomponent of the system that is faulty." Such a representation would be very valuable, because it would provide very clear statements of what the system is intended to do that the user could then compare with his own requirements to see whether or not the system is intended to solve a particular problem and how well it does that.

The experiment thus highlighted two fundamental limitations:

1. There was still no explicit representation of the intent behind a goal, which the system could use to explain *what* it was doing. Thus, the system could not explain what it meant to achieve a goal.
2. There was still no representation of relationships between problem-solving knowledge and domain knowledge, which could account for *why* a method worked.

The problem lay in the fact that the methods were themselves "compiled" from a still more abstract level of knowledge, but neither that knowledge nor the derivation of the plans from it was represented. We want to represent this specification-level knowledge explicitly and then mechanically derive the plan-level knowledge from it. More specifically, this involves the following activities:

```
To diagnose a decomposable system,
  If there is a fault in the system,
   then locate the cause of the fault within the system.

To diagnose a primitive system,
  If the system is faulty,
   then conclude it is the diagnosis.

To locate the cause of a fault within a system which is loosely-coupled,
  Diagnose the subcomponents of the system.

To locate the cause of a fault within a system which is tightly-coupled,
  Locate the cause of the fault along the signal-path
    beginning at the system-input and ending at the
     system-output.

To locate the cause of a fault beginning at system1 and ending at system2,
  If system1 is faulty
    then diagnose system1
    else locate the cause of the fault along the
     signal-path beginning at the system that system1
      outputs to and ending at system2.
```

Figure 2-3: Methods as an inadequate explanation of the goal of diagnosis

- **Develop a declarative representation of domain-descriptive knowledge that is conducive to transformation to problem-solving knowledge.** This representation consists of two components:

  1. *Definitions* of terminological entities
  2. *Assertions* of relationships based on those entities

- **Define a set of primitive actions for the description of goals.** It is assumed that these actions will be readily understood by users without explanation. We are finding it possible to characterize goal intent in terms of a small number of primitive actions. Thus far, we have found two:

  1. `determine-whether`: establish the truth of a given assertion
  2. `find`: find an object that satisfies a given description

  While we anticipate the possibility of other primitives, it is worth noting that these correspond to the basic types of problems analyzed by Polya [6]: problems to prove and problems to find.

- **Define a representation of domain-specific goals based on primitive actions.**

- **Represent "weak methods."** We wish to regard weak methods as problem-solving knowledge that is not specific to a problem domain but is

available when such specific knowledge cannot deal with a particular goal. With respect to the primitive actions just cited, our weak methods may be regarded as "first principles" that may be engaged in satisfying `determine-whether` and `find` goals, regardless of the specific nature of the assertion to be established or the description of the entity to be found.

- **Represent optimization knowledge.** This is the knowledge that makes the expert system efficient by turning "weak methods" into "strong" ones. It includes knowledge about how to make generate-and-test more efficient by moving tests into the generator. By representing this knowledge (and its application) explicitly, we will be able to answer questions such as why a particular test was performed at a particular time.
- **Create a reasoning mechanism capable of deriving procedural problem-solving knowledge from declarative domain-descriptive knowledge, the "weak methods," and optimization knowledge.** The desired explicit representation of the relationship between problem-solving knowledge and domain knowledge will be based on this mechanism. This mechanism will create plans that are stated roughly at the level of abstraction of the current plan representation in EES. Like our current plans, these plans will be used by the automatic programmer as it synthesizes an expert system. The difference will be that, because the derivation of these plans is recorded, the explanation facility will be able to explain how individual steps in the plans contribute to achieving a goal, and it will be able to further elaborate just what that goal means.

To us, the *really exciting possibilities for this research are that, in addition to* providing increased explanatory power, it will make explicit and allow us to exploit the link between the "weak methods" of traditional AI systems and the "strong methods" of expert systems. This should give us a better understanding of the nature of the knowledge in expert systems and how it is employed.

2.5.1.1 A hypothetical example

To illustrate the kinds of explanatory gains we expect, let us return to the digital circuit of Figure 2-1. In Figure 2-2 we presented an explanation that might be produced by paraphasing the code of an expert system designed to diagnose this circuit. This explanation is barely adequate, and certain important kinds of explanations cannot be provided:

- **Questions about terminology cannot be answered.** The system cannot answer a *question such as*, "What *does it mean for* adder A1 to be faulty?" because the definitions for such terms are not available.
- **Justifications cannot be provided.** The system cannot explain why it is important to look at A1-OUT1 in terms of the role it plays in the circuit being diagnosed, because the system doesn't know.
- **Questions about general methods cannot be answered.** Because all of the system's problem-solving knowledge is represented at a very specific level, there are no means to answer questions such as, "In general, how does the system diagnose a decomposible system?"

In Figure 2-4 we present the kinds of explanations that we expect to be able to produce using our enhanced specification language. The system will be able to present justifications that show why a particular circuit component is being considered in terms of the role that component plays in the circuit. It will also be able to answer questions about terminology and the meaning behind goals. Finally, because we will have a much better representation of the derivation of problem-solving knowledge, we expect to be able to provide better explanations of how the system's problem-solving knowledge works in general.

Justifications:

System: Please enter the value of the signal at A1-OUT1:
User: Why?
System: DIAGNOSER is trying to determine whether adder A1 is faulty. A1-OUT1 is the output port of adder A1. A system is defined to be faulty if its actual outputs are not equal to its expected outputs and its actual inputs are equal to its expected inputs.

In this case, it has been determined that the actual inputs to A1 are equal to the expected inputs, so if the value of the signal on A1-OUT1 is not equal to the expected value (which is 6) the system will conclude that A1 is faulty.

Questions about Terminology and Intent:

User: What is a decomposible system?
System: A decomposible system is a system that has more than 1 subcomponent.

User: What does it mean to diagnose a decomposible system?
System: To diagnose a system means to find a primitive subcomponent of the system that is faulty.

Questions about General Methods:

User: How does DIAGNOSER diagnose a decomposible system?
System: DIAGNOSER locates the fault along the signal path(s) beginning at the system's input(s) and ending at the system output whose expected value is not equal to its actual value.

Figure 2-4: Hypothetical explanations planned for EES

2.5.2 Operationalization

We would like to address two areas in improving operationalization: interpretation and compiled-in assumptions. These areas are explained in detailed below.

2.5.2.1 Interpretation

Currently, the EES framework compiles the specification for the expert system into an implementation. We would like to modify the framework to support direct interpretation of the specification as well as compilation. This would allow an expert system designer to more rapidly experiment with design changes and also would allow us to make use of compiled-in assumptions.

2.5.2.2 Compiled-in assumptions

It is well known that experts make assumptions. Sometimes an expert will explicitly make an assumption during problem-solving, but often assumptions are compiled into a problem-solving method so that, by adopting a particular method, the expert implicitly makes assumptions. For example, although a novice might methodically consider and reject very unlikely problems when diagnosing a device, a more experienced troubleshooter won't consider them, assuming that they aren't causing the problem because they are so improbable. He doesn't even appear to be consciously aware that he is making these assumptions. An intriguing possibility for EES would be for the program writer to compile such assumptions into the code it writes. This would make the program writer a non-equivalence-preserving compiler, since the deep specification knowledge would be assumption-free, while the compiled knowledge would contain assumptions. These assumptions would be explicitly recorded in the refinement structure so that they could be retracted later if they appeared to be wrong.

Making such assumptions could significantly improve the efficiency of an expert system in terms of both runtime and the amount of data for which it queried the user. There seem to be three major considerations in making an assumption: 1) it must be correct most of the time, 2) it must be possible to determine when it is violated, and 3) it must be possible to recover from an assumption violation. The benefit of this approach is that it would allow us to create expert systems that solve straightforward cases quickly, while still being robust enough to handle complex cases.

2.5.3 Explanations

To provide better explanations, explanation will be treated as a planning problem where the system plans an explanation to provide the user with the knowledge he appears to need to know. This approach is:

- Flexible: the system can represent many different explanation strategies and employ them as circumstances warrant.

- Easy to modify: strategies provide a more modular organization.
- Reactive to the user: if the user appears not to understand the explanation, the system will either replan a new explanation or interactively debug the current explanation.

REFERENCES

1. Clancey, W., "The epistemology of a rule-based expert system: A framework for explanation," *Artificial Intelligence* 20, (3), 1983, 215-251.

2. Davis, R., and J. King, *The Origin of Rule-Based Systems in AI*, Addison-Wesley, 1984.

3. Moser, M. G., "An overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt Beranek and Newman, Inc., 1983. BBN Technical Report 5421.

4. Mostow, J., and W. Swartout, "Towards explicit integration of knowledge in expert systems: An analysis of MYCIN's therapy selection algorithm," in *Proceedings of the National Conference on Artificial Intelligence*, 1986.

5. Neches, R., W. Swartout, and J. Moore, "Enhanced maintenance and explanation of expert systems through explicit models of their development," *Transactions on Software Engineering*, November 1985. Revised version of an article in *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, December 1984.

6. Polya, G., *How To Solve It: A New Aspect of Mathematical Method*, Princeton University Press, Princeton, New Jersey, 1971. Second edition.

7. Scott, A. C., W. J. Clancey, R. Davis, and E. H. Shortliffe, "Methods for generating explanations," in B. Buchanan and E. H. Shortliffe (eds.), *Rule-Based Expert Systems*, Addison-Wesley, 1984.

8. Swartout, W., *A Digitalis Therapy Advisor with Explanations*, Massachusetts Institute of Technology, Laboratory for Computer Science TR-176, February 1977.

9. Swartout, W., "XPLAIN: A system for creating and explaining expert consulting systems," *Artificial Intelligence* 21, (3), September 1983, 285-325. Also available as USC/Information Sciences Institute, RS-83-4.

10. Swartout, W., "Knowledge needed for expert system explanation," *Future Computing Systems*, 1986.

11. Swartout, W., and R. Neches, "The shifting terminological space: An impediment to evolvability," in *Proceedings of the National Conference on Artificial Intelligence*, 1986.

12. Teach, R. L., and E. H. Shortliffe, "An analysis of physicians' attitudes," in B. G. Buchanan and E. H. Shortliffe (eds.), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Mass., 1984.

3. FORMALIZED SYSTEM DEVELOPMENT

Research Staff:
Neil Goldman
Tom Kaczmarek
Don Cohen
Michael Fox
Kirk Kandt
Jay Myers
Kai Yue

Support Staff:
Audree Beal
Jeanine Yamazaki

3.1 PROBLEM BEING SOLVED

The Formalized System Development (FSD) project is working to extend the currently available technology in two areas of computer science: interactive *programming environments*, and the use of *specification languages* in software development.

The term *specification language* is not used in a technical sense in the computer science literature. It is used descriptively, in contrast to *programming language*, to refer to formal languages whose semantics can be defined in terms that tend toward the "declarative" rather than the "procedural." Statements in specification languages give the feeling of defining "what to do." Statements in programming languages give the feeling of defining "how to do it."

Ideally, a specification should serve at least two purposes. First, it should serve as a "contract" between the system specifier and system implementor(s), delimiting the permissible behaviors for the implemented system. Second, it should serve as a major portion of the "official" documentation of the implemented system.

In current practice, formal specifications are used in software development only when they are required by the government or management as documentation, or when there exists a fully automatic means of implementating the specification, i.e., a *compiler* for the specification language. The first use is dangerous, because there is no effective means of keeping the specification consistent with an evolving, manually produced implementation. Hence the documentation becomes obsolete and misleading. The second--operational use of formal specifications--is quite limited, because even the most high-level, general-purpose languages for which compilers exist [7, 13] lean heavily toward the "programming language" end of the specification-programming spectrum. Specification languages are in real use only within quite restricted problem domains for which special-purpose languages and compilers have been designed [14, 15].

The primary reason for relying on automatic compilation of specification languages is the fact that most software is intended to evolve over a lifetime of many years. For the specification to serve the two primary roles mentioned above, it must be the *specification itself* that evolves with the implementation (and with any derivative documentation), tracking that evolution. This is currently practical only if a new implementation can be derived automatically from an altered specification.

Recompilation of a large system, even when automatic, is an onerous and time-consuming task. Since changes to a specification typically affect a small percentage of the entire system, considerable effort has gone into providing developers with *programming environments* [6, 9, 11, 12] that are able to analyze software interdependencies sufficiently to permit *incremental* recompilation of systems. Because automatic compilers currently perform almost no global optimization, this dependency analysis does not need to be very sophisticated. Programming environments not only provide for incremental recompilation, but also provide a housing for a variety of tools that manipulate or reason about programs. Equivalent support must be available to support the evolution of *specifications*.

Considerable research has been done in the area of specification languages [4, 8, 10]. Much of this work begins with predicate calculus as a logical base for specification, often augmented with some form of temporal logic to deal with the inevitable notion of *state* that arises in specifying software systems. Our work has focused on Gist [1] as a specification language. Gist has much in common with the logic-based languages, but it advocates a mixture of procedures and constraints as the means for denoting the desired system behavior.

High-level specification languages do not lend themselves well to automatic compilation. One reason for this is the need for considerable theorem-proving to discover correct implementations. Another is the existence of an extremely large number of potential (even correct) implementations, most of which are inadequate on efficiency grounds. Finally, these languages do not contain sufficient information upon which to base a choice among alternative implementations, because issues such as relative frequency of execution of various operations, or environmentally imposed resource restrictions, are not expressible within the language notation.

3.2 GOALS AND APPROACH

The goal of the Formalized System Development (FSD) project is to provide order-of-magnitude improvements in the cost of software development and maintenance through the use of a new software lifecycle paradigm. The new paradigm calls for the mechanical, but human-guided, derivation of software implementations from *formal specifications* of the desired software behavior. It relies on altering a system's

specification and rederiving its implementation as the standard technology for software maintenance [2]. Practical use of this paradigm requires extending active machine involvement into the earliest stages of the development process.

The specification-based software paradigm (see Figure 3-1) enables system builders to separate concerns, methodologically and notationally, that are not separable in the paradigm prevalent today. Specifically, issues of *functional behavior* of a system can be expressed and explored without the necessity for considering any specific *implementations* that produce the behavior. Implementation concerns can be expressed, and alternative implementation strategies explored, in the context of a fixed specification. This separation of concerns is possible today only with the use of *informal* specifications. However, informal specifications are susceptible to multiple interpretations, to misinterpretation, and to errors of omission. But most significant is the fact that they are not interpretable *by machine* at all. This prohibits the development of software aids for validation, implementation, or evolution of the specifications.

If system builders are to derive the full benefit of the new paradigm, support must be provided for the development and maintainance of formal specifications. This support entails the use of suitable languages (notations) for system specification, a methodology for using specifications, and suitable software support environments to assist people in their use. A minimal, but nonetheless very useful, environment can be patterned after existing programming environments for high-level programming languages [6, 9, 11, 12]. But these environments rely to a great extent on the ability to compile and execute pieces of a program (specification) on test data as a means of determining the correctness of that program. The use of higher level specifications does not, however, necessarily reduce the cost of producing a testable implementation. To be directly testable, a specification must be automatically machine executable (compilable). This is at odds with having a notation that supports evolution and validation techniques well. The FSD paradigm relies on *rapid prototyping* [3] as a means of reducing the effort needed to produce testable implementations of formal specifications.

Since producing an adequate implementation of a specification is expected to be more costly than producing the specification itself, there is a large potential benefit from technology that can aid in the validation of a specification *prior* to the investment in a testable implementation. When the specifications are *formal*, the opportunity is present to apply Artificial Intelligence and software engineering techniques to the problem of validation. In particular, *symbolic evaluation* [5] can be performed on the specification itself to provide feedback to the specifier on implications of a given specification.

The FSD project is engaged in the construction of a testbed for exploration of this

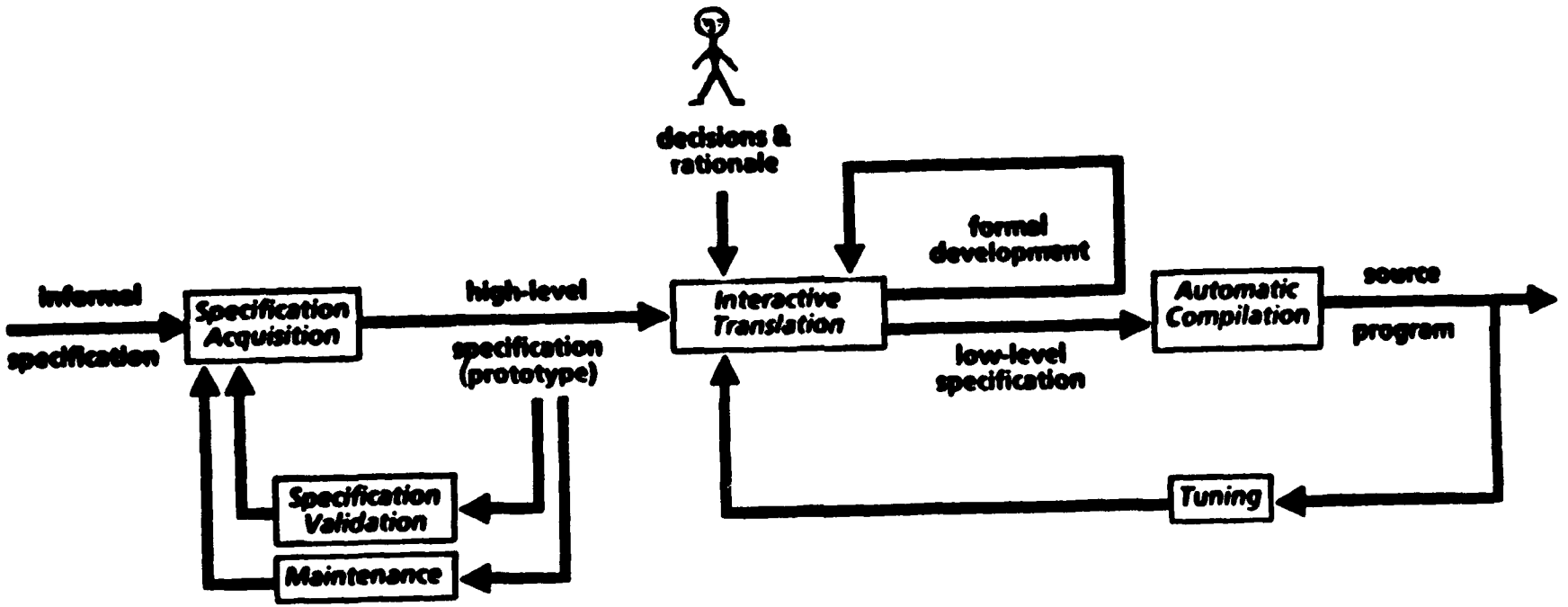


Figure 3-1: Formalized System Development methodology

new paradigm. The testbed constitutes an operational support environment for software design, implementation, and maintenance. Over the project's lifetime, this environment will support a succession of specification languages and implementation technologies. Progress is measured by increased support for the software designer in *formulating*, *validating*, and *implementing* software systems. The testbed focuses on supporting a level of support adequate for constructing useful *prototypes* of realistic software systems.

Each stage of the testbed system is being built using the prototyping environment supported by the prior stage. In this way, the project achieves leverage from its own research. In addition, several administrative applications are being maintained using the latest testbed support as an ongoing means of obtaining early feedback on the utility and quality of new support technology and tools.

3.3 SCIENTIFIC PROGRESS

Progress in the FSD project is best characterized in terms of its level of specification support, testbed implementation, and experience gained using the specification-based paradigm.

3.3.1 Host Language Integration

Currently FSD supports an annotation-based specification paradigm. This is achieved by extending Common LISP with specification-oriented macros and run-time library, and providing annotations (compiler pragmas) that control the translation of these macros into Common LISP code and invocations of the run-time library routines.

The language of extensions (both macros and annotations) is called AP5. It is not feasible to guarantee the existence of a set of supported annotations that will turn a given problem specification into an acceptably efficient implementation. For the annotation paradigm to be practical, there must be a way around this limitation. For AP5, the means we have chosen is to tightly integrate the specification with its lower level embedding programming language, Common LISP. This makes it possible to use Common LISP directly where the annotation language is inadequate, or where the specifier actually finds Common LISP more expressive than AP5. In addition, it makes it possible for AP5 to ignore certain facets of program behavior (such as I/O), allowing them to be handled entirely by constructs from the embedding language.

This tight integration manifests itself through the following:

- Support for the use of any LISP datum as an element of a tuple in AP5's database. Thus the specification is not restricted to referring to relationships between datatypes defined by AP5 or within the specification itself.

- Support for the use of any of the predefined Common LISP notions of equality (EQ, EQL, EQUAL, STRING-EQUAL, etc.), or other equivalence predicates defined by the user. This support means that applications, as in Common LISP, can use different theories of equality for different uses of different data. The specifier declares (or defaults) a particular equivalence relation independently for each slot of each relation.
- A natural mixture of AP5 and Common LISP in the specification text. In particular, AP5's predicates can be used anywhere within a LISP form as an expression that will evaluate to T or NIL. Conversely, within a primitive wff in AP5, it is possible to use an arbitrary LISP expression wherever a literal could appear.[1] Such expressions are evaluated relative to the lexical environment in which the embedding wff occurs. Iteration is specified in a syntax that extends the loop macro, with the multiple values generated being passed out of the database into Common LISP variables.
- Support for an arbitrary Common LISP predicate of n arguments to be treated as an n-ary relation, and an arbitrary Common LISP function of i inputs and o outputs to be treated as an i+o-ary relation. This is valuable not only in situations where first-order logic is inadequate for defining such relations, but for interfacing AP5 specifications to independently written Common LISP programs.
- Support for the full power of Common LISP's macro-defining capability for user extensions to the syntax of wffs.

One of the advantages of a database-oriented prototyping language is that it enables application programs to be integrated at a shared-data (rather than shared-file) granularity. Currently only multiple applications on a single workstation are being considered. Since these applications may run asynchronously (as separate processes sharing the same virtual address space), suitable locking of database transactions was provided. This was accomplished by "overloading" the semantics of AP5's atomic database transaction operator. A process entering an atomic transaction obtains a "lock" on the entire database. Another process may read the database without holding the lock, but it may not enter an atomic transaction of its own. The process holding the lock gathers data from the database and determines desired changes. After these changes have been approved--and perhaps augmented--by AP5's consistency manager, the process performs the actual update of the representation of changed relations while holding a second lock.[2]

---

[1] Or even where the name of a relation could appear, just as is done with FUNCALL and APPLY in Common LISP.

[2] Because the time needed to update the stored representations is generally quite small, this second lock has actually been implemented with a "critical section" of code during which process switching is inhibited.

3.3.2 Testbed Experience and Implementation

In order to benefit from most of the environmental (as opposed to language) support offered by the FSD testbed, application programs must be represented and maintained in the testbed's fileless environment. Existing applications, on the other hand, including the testbed implementation itself, are written as collections of text files. To bridge this gap, a source code "importer" was written to convert existing file-based applications to testbed-resident applications. The importer converts a LISP source code text file into a testbed module. The importer has the following salient characteristics:

- Each top-level form in the file becomes a single individual definition, which is made a component of the module.
- Commentary text appearing in the text file between top-level forms is retained as the value of the DOCUMENTATION attribute of the definition created for the top-level form following the commentary.
- The importer recognizes the syntax of standard defining forms of Common LISP (defun, defstruct, defvar, deftype, etc.) and is able to automatically classify the definitions it creates as *Function-Definitions*, *Variable-Definitions*, etc. It is also able to choose a suitable name for the definition by using the name of the Common LISP unit (function, variable, type) defined by the form.
- The importer recognizes occurrences of EVAL-WHEN forms at the top level of text files. Each form within the scope of the EVAL-WHEN is converted to a separate definition. The eval-when times list is saved as an attribute of each of these definitions.
- The importer recognizes uses of IN-PACKAGE at the beginning of files, and retains this information in the READ-PACKAGE attribute of the module being created. Subsequent IN-PACKAGE forms cause READ-PACKAGE attributes to be asserted on the definitions created for the applicable forms.
- The importer retains the total ordering of forms from the text file as the LOAD-ORDER attribute of the new module.

The importer has made the job of transferring existing LISP software into the testbed relatively painless. The recommended methodology is to create a *Development-Free-Module*, import each file of the target application program as a submodule, add any additional structure desired,[3] and simplify the load-order. Then the root module is converted into a System, which can be maintained with the testbed technology.

Experience with the testbed has consisted of maintenance of portions of the testbed itself and of several administrative software applications, such as electronic mail and

---

[3]Typically, testbed systems are broken into modules of smaller grain size than text files. Also, components can be placed in more than a single module, providing multiple organizations of a single system.

document preparation. A major limitation demonstrated by this experience is that small changes to relatively stable systems are very expensive to make, because it is necessary to map the entire persistent representation of the system into the virtual database in order to make any changes. A solution is planned that will permit the selective incorporation of individual components of a system.

One of the main impediments to using the testbed's software development monitor in the past had been a lack of reliability in correctly restoring the state of software systems and development histories following workstation initialization. These problems have been traced to lack of adequate separation of concerns about execution, definition, and development history environments in the testbed database. A design has been drawn up, and is being implemented, that calls for recording of update modes and markers for each of these aspects for each system in a workstation. This should not only relieve the existing reliability problems, but increase users' flexibility in restoring only needed portions of an environment from the persistent to the virtual database.

The other major dissatisfaction with the environment deals with overall performance. Some of these issues can clearly be dealt with by improved choice of annotations. In other cases, the real problem is not performance per se, but a problem of priorities--the software development monitor is competing with the developer for scarce computing cycles. Where possible, we want to give the developer priority, letting the monitor use spare cycles running as a background task.

The testbed was fully converted to run on LISP workstations (Symbolics 3600 series and TI Explorers) without reliance on the Interlisp Compatibility package. Reliance on Zetalisp is also being removed so that, to the extent possible, the testbed will be a portable Common LISP software system.

3.4 MILITARY IMPACT

Revolutionary advances in hardware technology over the past decade have outpaced those in methodology and computer support for the design of software. While more powerful hardware makes possible some savings in software development time and cost by reducing the need for optimization, these savings are relatively small. Software development and maintenance expenditures estimated in excess of twenty million dollars daily are ample evidence of the economic burden to DoD. Managerial controls can help to ensure that software funds are spent effectively, but they cannot stem these burgeoning costs as they are inherent in the current complex, manpower-intensive software production and maintenance technology.

For many military objectives, the complexity of software is an even more important limitation on realizable systems than their financial cost. The new hardware

technologies provide the basis for far more complex and evolvable software, in both embedded and non-embedded applications. Such software cannot be reliably produced by current software methodologies, which were developed in a world of hardware resource scarcity for both the developer and the ultimate delivery system. These hardware technologies--VLSI, massively parallel machines--will rapidly become available throughout the industrialized world. Advances in software technology propagate more slowly. It is critical to DoD's mission that the U.S. retain its leadership role in this area and become the first to effectively tap the potential afforded by the ongoing hardware revolution.

The FSD project addresses these challenges by providing the framework for the use of formal specifications as the cornerstone of an improved software development paradigm [2]. This paradigm will dramatically improve military programmers' ability to develop and maintain large software systems.

The most promising strategy to control software costs and to realize the ever-increasing potential of new hardware is automation of the development and maintenance activity. FSD proposes to automate significant portions of the current programming methodology, shifting the focus of human effort to making specification and strategic implementation decisions. The targeted specification language of this effort, Gist, is oriented toward specifying total systems, not just software components. Gist specifications encompass systems that are ultimately implemented with combinations of hardware (both electronic and mechanical), software, and "fleshware" (human) components. This is particularly relevant to the heavy use of "embedded" systems in military applications.

3.5 FUTURE WORK

A technology to support specification-based design, implementation, and maintenance of software systems cannot first be fully designed, and then implemented, with any chance of success. Rather it must be evolved through feedback gained from actual use of a succession of operational environments that can span the gap between current technology and the goal. Because of this, and because we believe this is the way almost any large software system should be developed, we have focused on supporting *rapid prototyping* in this environment. We now have in place a robust, well-exercised testbed that supports the specification-based paradigm. This enables us to gain leverage from each successive environment in its own maintenance, as well as in specifying and implementing its successor.

ISI also has in place a technology transfer program (see Chapter 1) that enables us to disseminate this technology to other researchers in the DARPA community. This will provide our major source of feedback, both on the extensibility of the testbed and on its

strengths and weaknesses in use on serious application development and maintenance. Such feedback is essential to gain maximum leverage from the FSD's future research and development efforts.

The weakest aspect of the existing testbed is the specification language itself, which is an extension of Common LISP. The anticipated benefits from the new paradigm are predicated largely on gains derived from maintaining much higher level specifications. Having categorized specification language features and the benefits afforded by each, having designed a succession of languages that incorporate these features, and having laid out the framework needed for implementation support for each language level, we are prepared to make a series of quantum steps in the specification technology supported in the testbed.

Progress in the operational testbed will be characterized by distinctive plateaus, each enabling behavioral specifications to be written with less regard for the ultimate implementation hardware/software base, and each with suitable support for deriving implementations from those specifications for (at least) a Common LISP software base and the supported LISP workstation. This support will extend to maintenance of specifications by multiple maintainers and distribution of software upgrades to multiple users, at least where the maintainers and users are connected by a local area network.

The precise character of each plateau must be in some significant ways opportunistically determined by feedback from users of its predecessor. We expect to achieve the plateaus characterized below during the next three years. We recognize that some features we use to characterize a plateau are independent of other features of that plateau, and logically could be present in an earlier plateau. The decision of when to target the introduction of a new capability is based on the need for coresident capabilities and on our estimation of the technical difficulty of providing that capability from the current state of the art.

- At the *local annotation* plateau, specifiers will be able to use a pure extension of Common LISP as their specification language. This extension will permit them to omit, selectively or totally, any specification of target language data structures to be used, and to specify logical tests and data retrievals over their data with full first-order logic, without regard to their procedural realization. The specification language will also provide for the expression of logical consistency conditions that must be maintained by the implementation, and for data-driven invocation of procedures (automation rules). Annotations added to these specifications will guide an automatic compiler in producing pure Common LISP code that implements them. *Human implementors will select annotations* with some knowledge-based assistance. This assistance will include, but will not necessarily be limited to, monitoring annotations that must still be added to make the specification compilable, providing menus of annotations that are applicable to selected parts of the specification, and prohibiting the selection of incompatible annotations.

- At the *independent specification language* plateau, the specifier will be using a notation that is no longer an extension of the target programming language, and that does not rely on the use of target language procedures for parts of the specification. This language will have a grammar-defined syntax. Functional and procedural abstractions will at least permit, and possibly require, typed inputs and outputs, to support greater analytical support at the specification level. The language will have an extensible typing mechanism and multiple inheritance. Implementation will still be by annotations to the specification, but many annotations will have the form of transformations, making it possible to add new annotations without altering the compiler. The power of the specification notation will still be compromised sufficiently to permit treatment of the annotations as an unstructured collection of pieces of advice.
- At the *implementation design* plateau, the specification notation will be sufficiently abstracted from the implementation hardware architecture that annotations treated as advice in terms of the specification itself will not yield acceptible implementations, probably even for prototyping purposes. The implementor, with machine assistance, will have to provide the compiler with a derivation plan for arriving at an effective implementation. Such a plan will be much like a conventional program, whose data is the specification to be implemented and whose operations are transformations on that specification. The implementor will be supported in his search for a suitable derivation plan by automated management of alternative partial plans. He will be able to selectively extend these plans and, in some cases, merge plans. The ability to reuse most of a derivation plan when an altered specification must be reimplemented is crucial to the viability of this plateau.
- At the *evolvable specification* plateau, support for performing maintenance on the specification itself will go beyond what can be provided solely by knowledge of the syntax and semantics of the specification language. This additional support may entail the specifier providing additional information about the specification, such as *purposes*, *preferences*, and *scenarios*. Our goal is to allow the specifier to describe desired changes in terms that are on a higher level than localized syntactic changes to the specification. Examples of higher level editing terms include the following:

  - changes described as exceptions to the current specification
  - changes described in terms of the set of behaviors denoted by the current specification
  - changes that cross the syntactic boundaries of the language, such as further "parameterizing" a current abstraction, or "generalizing" a current abstraction

  ISI is already engaged in research on specification evolution, and notations to support it, under a separately funded contract.

As each plateau is achieved, necessary changes will be made to the existing specifications of the FSD administrative applications, and the FSD testbed software itself, so that the continued development of these systems will take place within the environment provided by the new plateau.

Several aspects of lifecycle support require forms of dependency tracking. Specification maintenance is simplified if the specifier can be automatically notified when a change to one part of his specification requires that he modify a dependent piece. Incremental compilation is possible only if the dependencies that connect implementation code to the original specification are maintained. Specification-level explanation of implementation behavior can only be realized if these dependencies are available to the explainer. These are not dependencies that can be expressed in the logical formalism used for FSD consistency maintenance. We need to find a general mechanism for expressing and tracking these dependencies, rather than implementing ad hoc mechanisms for each new kind of dependency that arises.[4] It is very often desirable for a software system to be able to operate in a number of usage environments. The differences may be in implementation hardware, peripherals, associated software (such as an operating system), or user community. The ramifications of these differences may be confined to the implementation of the specification, or may engender specification differences as well. For maintenance purposes, it is unacceptable for each such "end version" to be managed as an independent specification with its own implementation. We propose to support the concept of a "version dimension" with an enumerated collection of values. For example, PRIVACY, with values PUBLIC, PERSONAL, and SENSITIVE, might be a dimension along which a specification was differentiated, while HARDWARE, with values SYBOLICS-3600 and TI-EXPLORER, might be a dimension used in differentiating implementations. We want to make it possible for a specification and implementation plan to have a "generic" version, with differences specified along these dimensions only where needed.

Managing changes to specifications and their implementations, and distributing software upgrades to users, place developers in a situation of constantly shuffling between partially completed tasks, with many activities pending the outcome of others in progress.[5] We believe that the automation rule component of FSD's operating system places us in a particularly strong position to automate some of this agenda management.

Having a high-level, formal specification of system behavior, together with a record of the mapping from that specification to its implementation, opens up the possibility of creating software that is tailorable by end-users in ways that go well beyond what can be achieved with canned "user-model" parameters. Particularly promising are the possibilities of letting end-users add modeling extensions to a system, where these are

---

[4]A similar problem was recognized in AI knowledge bases that perform forward inference, thereby storing results whose logical support could later be withdrawn. This problem is dealt with by a technique called truth maintenance.

[5]This situation is not unique to software management, of course.

defined in such a way that they cannot be inconsistent with the specification, nor invalidate its implementation, and letting them add personalized automation rules to their own environments. We have already seen this occurring with the software-knowledgeable end-users of our existing administrative applications.

References

1. Balzer, R., Donald Cohen, M. Feather, N. Goldman, W. Swartout, and D. Wile, "Operational specification as the basis for specification validation," *Theory and Practice of Software Technology* SE-8, (1), 1983, 21-49.

2. Balzer, R., "A 15-year perspective on automatic programming," *IEEE Transactions on Software Engineering* SE-11, (11), November 1985, 1257-1268.

3. Balzer, R., N. Goldman, and D. Wile, "Operational specification as the basis for rapid prototyping," in *Proceedings of the Second Software Engineering Symposium: Workshop on Rapid Prototyping*, ACM SIGSOFT, April 1982.

4. Burstall, R. M., and J. A. Goguen, "Putting theories together to make specifications," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 1045-1058, IJCAI, 1977.

5. Cohen, Donald, "Symbolic execution of the Gist specification language," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

6. Kernighan, B. W., and J. R. Mashey, "The UNIX programming environment," *IEEE Computer*, April 1981, 12-24.

7. Liskov, B., et al., *CLU Reference Manual*, Springer-Verlag, Berlin, 1981.

8. Manna, Z., and R. Waldinger, "A deductive approach to program synthesis," *ACM Transactions on Programming Languages and Systems* 2, (1), January 1980, 90-121.

9. Osterweil, L. J., "Toolpack--an experimental software development environment research project," in *Proceedings of the Sixth International Conference on Software Engineering*, pp. 166-175, IEEE Computer Society, Tokyo, September 1982.

10. Scheid, J., and S. Anderson, *The Ina Jo Specification Language Reference Manual*, System Development Corporation, Technical Report TM-(L)-6021/001/01, March 1985.

11. Swinehart, D. C., P. T. Zellweger, and R. B. Hagmann, "The structure of Cedar," in *SIGPLAN 85: Language Issues in Programming Environments*, ACM SIGPLAN, July 1985.

12. Teitelman, W., *Interlisp Reference Manual*, Xerox Palo Alto Research Center, 1983.

13. Wegner, P., *Programming with Ada: An Introduction by Means of Graduated Examples*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

14. Wolverton, V., IBM Personal Computer VisiCalc. Personal Software, Inc., 1981.

15. Zave, P., "An operational approach to requirements specification for embedded systems," *IEEE Transactions on Software Engineering* SE-8, (3), May 1982, 250-269.

4. COMMAND AND CONTROL COMMUNICATIONS

Command Graphics:
Richard Bisbey
Dennis Hollingworth
Danny Cohen
Jan Brooks

Internet Concepts:
Jon Postel
Paul Mockapetris
Danny Cohen
Annette DeSchon
Greg Finn
Joyce Reynolds
Ann Westine
Gerard Parr

Multimedia Conferencing:
Steve Casner
Ehoud Haberman
Dave Walden
Alan Katz
Greg Finn
Brian Hung
Paul Mockapetris

Supercomputer Workstation Communication:
Steve Casner
Alan Katz
Annette DeSchon

4.1 INTRODUCTION

The Communications Research group is developing new applications for computer networks in a wide range of areas including Command and Control Graphics (C2G), Internet Concepts (INC), Multimedia Conferencing (MMC), and Supercomputer Workstation Communication (SWC). The interaction of these activities provides valuable insights into problems and potential solutions in communications research. Therefore, it is appropriate to report the projects of the Communications Research group together.

The Command and Control Graphics project has developed a network-distributed, display-device-independent graphics application for command and control. The graphics support system provides the ability to trade off the level of graphics interaction against the available network communication capacity.

The Internet Concepts project focuses on providing appropriate and effective designs for the primary user-service applications in the internetwork communication environment. Currently the focus is on the Domain Naming System.

The packet voice and packet video capabilities demonstrated on the Wideband Network and the mixed text, graphics, and voice messages demonstrated in multimedia mail are being combined into an interactive conferencing system by the Multimedia Conferencing project.

The Supercomputer Workstation Communication project began in October 1985 to extend the application of the Wideband Network to more traditional data traffic in addition to real-time traffic. The current project has developed routing methods to allow traffic that would normally flow over the ARPANET to be directed over the Wideband Network instead. Using medium-sized computers and workstations, data transmission rates many times the maximum possible on the ARPANET have been demonstrated. The Wideband Satellite Network provides the opportunity for effective remote access to supercomputer facilities.

4.2 PROBLEMS BEING SOLVED

The goal of the Communications Research group is to extend and enhance computer communications. The group's work is based on the ARPA-Internet, a system of interconnected digital packet networks for computer communication. Rules for communication (protocols) are the key element in successful computer communication. The ARPA-Internet has working protocols for communication between heterogeneous computers via an interconnected collection of heterogeneous packet networks.

This research covers work at several levels, involving applications protocols as well as host-to-host and gateway-to-gateway protocols. At the applications level, the focus is on new uses of the Internet based on new procedures for hierarchical naming, multimedia mail, and computer mail interoperation. The basic protocols are largely complete, but a number of extensions are being explored to integrate and extend the packet network technology.

The communication research group specializes in prototype development of new applications of computer networks. In particular, we have developed a device-independent, network-based graphics application-support system, and digital packet voice and video applications.

4.2.1 Command and Control Graphics

The military, like the private sector, is in the midst of an information explosion. More and more computers and computer-controlled systems are being acquired, generating information in increasing quantity and detail. For this information to be useful in command and control decisions, it is necessary for computers to assume a greater role in the storage, retrieval, analysis, integration, and presentation of data. Information must be presented to the decision maker in ways that enhance and facilitate the decision process.

A common aspect of virtually all command and control decision activity is the requirement to examine specific command and control data in an appropriate

geographic context. The proper presentation of such information on a background map can significantly aid in the rapid assimilation of information by the decision maker. Two-dimensional geographic presentations often disclose perspectives that would not be readily apparent from a table or list of numbers, providing a natural medium for integrating and fusing information. For example, a situation display might graphically integrate surveillance, force, and meteorological data reported by dispersed forces to produce an aggregate picture of a particular situation. The resulting graphics display could then be used to coordinate the activities of the dispersed forces. In general, C2 applications such as battlefield management, aircraft recovery, sensor avoidance, and retargeting are much easier to comprehend when presented in a geographic context.

Computers and computer automation are finding increasing utility in such C2 applications. However, effective command center use of computer-based C2 information requires much more than just computerization of the data. It also requires a suitable presentation mechanism for information display, an intelligent front-end to manipulate and present appropriate data geographically in an intelligible fashion. Consequently, for computer graphics to play a significant role in military decision making, three basic components are needed: basic graphics support; high-level, domain-independent display agents; and high-level, domain-dependent decision aids.

First, there must be a graphics system, a program that converts basic graphics primitives (e.g., lines, text, filled solids) into the order codes for a particular display device. The graphics system must meet military requirements for mobility and survivability, particularly in crisis mode. The system must also be adaptable to the changing communications, processing, and display resources available during and between crises, and must evolve to meet future command and control processing and display requirements.

Second, high-level, domain-independent display agents are needed. Display agents are "expert systems for producing graphics displays," i.e., they manage the display surface, the graphics representation, and the placement of information on a graphics display. For example, a display agent may know how to create and label a graph from a list of numbers or how to create a Mercator Projection map and draw and annotate a great circle course on that map given a list of geodetic coordinates. Display agents enable the user to create complex two-dimensional graphics displays easily, thereby removing the burden of "graphics expertise" from the user. Display agents use a graphics system as the underlying graphics support mechanism for displaying visual information.

Third, high-level, domain-dependent applications and decision aids in the form of expert systems are needed. These applications and decision aids interact with the user, the user's databases, and the display agent. The decision aids retrieve raw data and then generalize, summarize, analyze, abstract, aggregate, and fuse the raw data,

transforming it into information relevant to the user and the user's decision-making process. The decision aid also performs the semantic binding between the user's information and the high-level representations provided by a display agent. For example, a decision aid would determine the information relevant to the user (e.g., a threat envelope, a convoy of ships, a returning aircraft, or a recovery site) and then select the high-level display agent representations to be used to denote that information (e.g., a transparent overlay or a labeled icon), providing the geographic coordinates of the objects for placement on a map or chart. The decision aid would not be concerned with the particular techniques employed by the display agent or the graphics system to support the various representations, nor with the production of those representations.

4.2.2 Internet Concepts

This project is concerned with the ongoing development of various prototype implementations of new ARPA-Internet services (e.g., SMTP Text Mail, MPM Multimedia Mail, the Domain Naming System, Computer Mail Interoperations); in addition, it provides research studies of various computer communication issues.

The growth of the Internet has led to a problem in maintaining the database of names of computers (host names), their addresses, and other related information. The Domain Naming System is a replicated, distributed database system developed to provide a basis for distributed management and maintenance of this important host name database.

The usefulness of computer mail or electronic mail in the DARPA community and in the commercial world has led to a desire for some form of interoperation between the ARPA-mail world and some commercial mail systems. The Intermail program is an experiment in supporting such interoperation between the incompatible procedures of the commercial mail systems and the ARPA-mail world.

Another concern raised by the growth of the Internet is the development of appropriate routing procedures for very large systems. Most of the routing procedures use table space or information exchanges that grow with the size of the system (some grow in both ways). For very large systems, such procedures become unworkable. The development of the Cartesian Routing procedure is an exploration of a routing method that is independent of the size of the system.

The Internet Concepts project continues to assist in the ongoing evolution of the ARPA-Internet, especially in the introduction of new capabilities. This project aids with the coordination, assignment, and maintenance of network parameters such as network numbers and protocol identifiers, and with the development and maintenance of protocol specifications and other documents relating to the operation of the ARPA-Internet.

4.2.3 Multimedia Conferencing

The purpose of multimedia conferencing is to enhance the productivity of communications between individuals through the use of computers and networking. The DoD requirement to link all commanders with multimedia communication world-wide is a prominent example. Military commanders need a service that includes the following:

- Conventional communication, such as voice and video. In addition to support for packet-switched streams, the system should be capable of interfacing to the conventional phone system, video conferencing services, and other outside communication media. The ability to use different media is important, both to improve the speed and effectiveness of the cooperation, and also as a way of making computer-aided conferencing useful to people who are not computer professionals.
- Communication of computer-generated data such as maps, status displays, and text. The conferencing system is being designed to accept and distribute information from existing applications with minimal interfacing effort, and to provide for input and output of hardcopy text and images.
- A style that fits the needs of the commanders rather than requiring specialized computer knowledge. The commanders' expertise is our scarcest resource, and enhancing productivity is our goal. This includes the ability to organize information flow either as a peer-to-peer conference on a hierarchical basis, and to support separate styles for different missions, while preserving the desired information flows between separate conferences. For example, a weather service could have an internal flow of information between experts for different sensors, with the results edited and distributed to different force commanders according to their specific needs.
- An implementation that meets military requirements for survivability through redundancy, best possible operation in the face of resource loss, and preservation of interoperability with DoD standard protocol services.

The ISI Multimedia Conferencing project is directly relevant to these needs. Media protocols and support involve an area of effort that focuses on providing standardized formats and interfaces to devices for different media. For example, ISI has taken the lead in providing real-time video over packet-switched networks such as the Wideband Network.

The Multimedia Conferencing project uses components from multimedia and text mail tools produced by ISI and by others working in this area.

The architectural model for conferencing explicitly aims at producing tools for different conference styles, experimenting with prototypes, and providing solutions for applications. For example, we are developing support for democratic as well as authoritarian floor control during a conference, and we plan to experiment with information flow between conferences.

4.2.4 Supercomputer Workstation Communication

This project addresses the need for effective, high-bandwidth communication between powerful computers and their users. The rapid growth of the Defense Data Network (DDN) and the heavy load placed upon it demonstrates the demand for packet-switched data communication.

There are two major initiatives under way to provide significantly increased computing resources to research scientists in the military and in the DoD contractor community. One provides very powerful symbolic computers: this is a component of the DARPA Strategic Computing Initiative. The other is the NSF initiative on supercomputers to provide greatly increased numerical computation capability.

The objective of the Supercomputer Workstation Communication project is to provide effective remote access to these computing resources via high-capacity communication systems such as the DARPA Wideband Satellite Network. This provides several benefits of importance:

- High-bandwidth access to supercomputers at a limited number of locations will be available to researchers in DoD-sponsored programs at a larger number of remote locations, so that the power of supercomputing can be applied to problems that would otherwise have to wait.
- The protocols and routing methods developed to provide supercomputer access will also allow the Wideband Network to be used for some of the traffic now flowing over more heavily loaded terrestrial networks in the Defense Data Network.
- New techniques and applications of high-capacity computer communications technology developed by this project can be applied in military systems to be deployed in the future.

Thus, the SWC project will augment the DoD communication infrastructure in a general way while providing specific leverage for DARPA's existing commitment to supercomuting by making it more widely accessible.

4.3 GOALS AND APPROACH

The major effort for the Command and Control Graphics project involved the production of the Geographic Display Agent (GDA). The project incorporated a laser video-disk map database into the Geographic Display Agent and expanded the software map database and drawing algorithms to include state boundary information. This provided an improved map environment for the Geographic Display Agent and C3 applications by providing more context than the previous land mass outline maps. The project focused on supporting the Geographic Display Agent and Graphics System through the final Strategic C3 Experiment demonstrations. The Internet Concepts project's goals are to manage the assignment of protocol parameters to network

experimenters as needed, and to continue to evolve and incorporate further capabilities. The goal of the Domain Naming System effort is to produce up-to-date documents of the protocols used in the ARPA-Internet community on an as-needed basis and to investigate issues of addressing and routing in very large networks. The goals of the Multimedia Conferencing project are to develop protocols for conferencing management and media transmission, and to demonstrate multimedia conferencing by developing prototype systems. We have continued our efforts in the video and image media, and we have conducted weekly teleconferences to test the system's effectiveness and make improvements for higher quality video and sound. The goal of the Supercomputer Workstation Communication project is to create and improve communication between workstations and supercomputers, demonstrating first that traditional transport protocols can be used to communicate over the Wideband Network to a supercomputer on a local network connected to the Wideband Network, and then designing new protocols as needed for more effective communication.

4.3.1 Command and Control Graphics

The three basic components of a graphics aid for command and control (the graphics system, the display agents, and the applications/decision aids) form a natural layering, with the graphics at the bottommost layer and the application/decision aid at the top. During a previous contract, ISI focused on the bottommost level, the graphics system component. The effort resulted in the development of a network-distributable, display-device-independent graphics system for command and control. As part of that work, ISI also began research on the display agent and decision aids components, specifically, features required for the creation of geographic displays. The work resulted in the development of a Situation Display and Assessment application for retrieving and displaying Navy data.

Situation Display allowed an operator seated in front of one or more graphics display devices to pose natural language questions and commands to a distributed database manager and to receive natural language and graphics responses. Examples of natural language input included "Do any ships within 400 miles of Luanda have a doctor aboard?" and "Show me the destroyers whose radar is inoperative." Graphics responses could be in either of two forms, tabular or geographic. In geographic mode, ships and associated data (such as sensor envelopes and speed/course vectors) were automatically positioned on a background Mercator projection map. During the development of the Situation Display, three factors were found to be particularly important in producing aesthetic geographic-based displays: intelligent placement of objects, preservation of display precedence relationships between objects, and judicious color selection.

The placement of information on the display surface proves to be critically important if the display is to be readable. In creating maps and charts, cartographers go to great

lengths to improve readability, even to the extent of displacing objects from their actual geographic coordinates. For example, consider a map of the United States showing major transcontinental highways and railways. In many instances, the two run parallel to each other. If both the highway and the railway were represented at their exact geographic locations, the two would lie on top of each other, making it difficult to distinguish one from the other. A cartographer would displace one of the two some distance from the other so that both would be visible. In Situation Display, "symbol collision avoidance" algorithms were included to perform this same type of function.

Another important factor in display generation is precedence of overlays. It is often necessary to call attention to large areas. These areas might represent some meteorological phenomenon like cloud or fog cover; coverage by sensing equipment, such as a satellite footprint or a radar sensor envelope; or a path, such as an airroute or a sealane. While it is important to be able to distinguish visually the area in question, it is also important that the area be displayed in such a fashion as not to obliterate or obscure other data displayed in that same area. Overlay precedence allows one to denote these areas by making small changes to either the intensity or the chromaticity of the objects (or portions of objects) being displayed in that area (similar to placing a light-colored piece of cellophane over the area). Situation Display used overlay precedence in displaying satellite footprints, sensor envelopes, and sealanes. For example, satellite footprints could overlay sensor envelopes, which could in turn overlay ships, sea, or land. All were discernible.

Color also had important use in Situation Display. Color selection provided discrimination between nationalities for ship information. Highlighting (using blinking or color intensification) was used to draw special attention to individual ships, e.g., for responding to questions such as, "Of the ships being displayed, which ships have an ASW capability?" Colors and intensities were chosen so as not to unduly emphasize one type of object over another. Colors for background information were chosen to produce a visually apparent background.

Two important observations were made from the Situation Display development effort. First, the production of legible geographic-based displays requires a considerable amount of graphics intelligence and sophistication. In fact, the amount of programming necessary to incorporate the "graphics intelligence" often exceeds that of the application program or decision aid itself. Second, the representation mechanisms required for the Situation Display were essentially generic in nature. For example, the same annotation mechanism used to represent the location of a ship or submarine in Situation Display could also represent the location of a tank or gun emplacement on an Army Tactical display or an aircraft being recovered in an Air Force Bomber Recovery display. While the graphics display portion of Situation Display was highly tailored to a particular application domain (pictures were described in terms of specific Navy objects, e.g.,

Vessel Control Numbers, Unit Identification Codes, Ship Classes), if the binding of semantic information to graphic representation was done in the application program, the same graphics display package could be used by a variety of applications.

Since the resolution of many C2 problems requires such an interface, ISI proposed producing a Geographic Display Agent (GDA) that assumed responsibility for the generation of geographic-based displays and was general enough to interface with many different application problems. The primary goal was to make it possible for C2 application programmers to use--at minimal effort--this geographic front-end with their C2 application software.

In order to realize this objective, we determined that the system should

- provide convenient access to assorted maps, eliminating the need for the application programmer to develop background maps or understand the mathematics of map projection.
- present a text-based interface that could easily be used manually or by any program that has text-output capabilities, such as those written in LISP, FORTRAN, or Ada.
- include primitives for the more useful generic cartographic operations, such as symbol generation, symbol conflict resolution, map transformations, overlays, and color handling.
- have a syntax for which the majority of parameters have assigned defaults that need not be included for normal picture descriptions.
- be built on top of the existing ISI Graphics System, which had already solved the problem of display-device independence and distributable operation over computer communication networks.

4.3.2 Internet Concepts

The long-term goals of the Internet Concepts project are to provide appropriate and effective designs for the primary user-service applications in the internetwork communication environment. The designs are based on a set of host- and gateway-level protocols that provide the full range of service characteristics appropriate to a wide variety of applications.

Our approach has been to pursue in parallel the abstract design and experimental implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions.

4.3.2.1 Hierarchical naming system

In a large internet system, the management of names becomes a complex task. The simple central table of names approach used in the past has become unwieldy. To remove the practical and technical constraints, to provide new functionality for new applications, and to provide for the continued growth of the Internet system, the Domain Naming System is being developed. This system provides structured names and allows subdivision of name management duties.

Our goal is to provide a design for a distributed system of domain servers that manage the access to a distributed database of names and associated data. Our approach is to create the design for the system and to review it with interested parties in the DARPA research community. As a result of feedback received from these reviews, the design will be modified, and we will implement a prototype domain server. Based on our experience with this prototype implementation, we will again modify the design and produce the final specification.

4.3.2.2 Computer mail interoperation

We are conducting an experiment in the interoperation between otherwise incompatible communication systems by developing a prototype service for transmitting computer mail between the ARPA-Mail system and a group of commercial mail systems (Telemail, MCI-Mail, and ITT Dialcom systems). This requires (among other things) development of techniques for sending computer mail to destination addresses that may not be allowed (by syntax or semantic checks) in the originating mail system.

4.3.2.3 Studies, surveys, and specifications

Although the Internet system is now operational, many potential extensions are possible, which have been discussed and desired but which have not been designed, implemented, or tested. Our goal is to study some of these issues and, when possible, to provide designs for their eventual development.

The areas of addressing, routing, and multiplexing are particularly subtle and require careful attention. We have concentrated on these areas and have explored many options in technical discussions and memos. Our approach is to develop an understanding of these technical issues and to advocate the inclusion of general-purpose support mechanisms in the protocol specification.

In the operation and the evolution of the Internet system, it is sometimes useful to survey or measure the implementation status and performance of particular protocols. As the Internet system evolves, and as flaws and ambiguities are discovered, specification documents must be upgraded.

4.3.3 Multimedia Conferencing

The Multimedia Conferencing project is building on the successful results of the precursor projects Wideband Communication (WBC) and Multimedia Mail (part of Internet Concepts).

Development of the Multimedia Mail system continued in the first part of the year. The goal of that work was to provide specifications for computer-oriented data structures in order to communicate various types of data in messages, including text, graphics, and voice. Our primary interest was and is in the communication mechanisms rather than the user interfaces.

There are three major areas of effort in the Multimedia Conferencing project: conferencing management protocols, media protocols, and demonstrations. Our goal is to develop and specify these protocols, then implement them in prototype conferencing systems to demonstrate the feasibility and effectivenes of multimedia conferencing.

1. **Conferencing Management Protocols.** An overall conferencing management architecture is being specified and developed in cooperation with other DARPA contractors. This task area includes concerns of general conference architecture, control protocols, synchronization, information distribution, multimedia datastream connectivity, information pre-delivery, conference entry and exit management, and user interaction.
2. **Media Protocols.** The individual media building blocks are being specified and developed. This task area includes the design and implementation of the individual media protocols and algorithms.
3. **Demonstrations.** The prototype conferencing system will be available for regular use by research groups that would otherwise have to travel for meetings. The feedback from users not involved in the implementation of the conferencing system will be most valuable, to learn how the system can be improved.

4.3.4 Supercomputer Workstation Communication

The Supercomputer Workstation Communication project is extending the application of the Wideband Network to more traditional data traffic, in addition to real-time voice and video traffic. Our approach is to take several steps in creating and improving the communication between workstations and supercomputers. While we focus on this pair of communicating partners, the approach is generic and the protocols developed could be applied to a wide variety of situations. Some of the traffic normally flowing over the ARPANET might be diverted to flow over the Wideband Network instead, providing the benefit of increased capacity while offloading the ARPANET. A key step is to develop routing methods that will allow appropriate types of traffic to be diverted.

Our first step is to collect the existing hardware and software building blocks and to

demonstrate that they can be assembled and used. The key element in this is the Wideband Network. This network has some interesting features (for example, streams) that differ from or go beyond those found in other networks, and we may find it useful or even necessary to use those features to support communication between workstations and supercomputers.

We will confirm that the traditional protocols (Telnet and FTP) can be used to communicate over the Wideband Network to one or more supercomputers. We expect these protocols to function as they are currently implemented; however, substantial performance improvements may be gained by tuning some of the protocol parameters.

Our second step is to move past the traditional remote access services provided by these protocols, and to explore more sophisticated forms of interaction between workstations and supercomputers. To provide full control of the bitmap-oriented workstation screen, it is necessary to move up several levels of abstraction to the window package interface. Many application programs are now written using the window package provided by the machine manufacturer or operating system vendor. We intend to develop a "network virtual window package" that will allow a computation site to call window package routines that communicate over the network and to execute them locally on the workstation.

Our third step is to approach interprogram communication at a much higher level of abstraction. We believe that it ought to be possible to configure multimachine cooperating computations by connecting the outputs of one program to the inputs of another program. There are, however, many obstacles to putting this simple notion into practice--for example, format differences, timing dependencies, different parameter-passing methods, and differing levels of data abstraction. We will explore these obstacles and attempt to develop methods to overcome them.

4.4 SCIENTIFIC PROGRESS

The Command and Control Graphics project has created a Geographic Display Agent with an object-oriented interface. This system is composed of the Graphics System, the Display Agent, and the Map Manager. The Internet Concepts project has implemented the Domain Naming System and extended the Intermail service. The Multimedia Conferencing project has developed a real-time conferencing system including packet video, packet voice, and the use of multimedia mail capabilities in a conference context. The Supercomputer Workstation Communication project has participated in the upgrade of the WBNET, developed network monitoring procedures, and performed performance tests.

4.4.1 Command and Control Graphics

A task was initiated to design and develop a high-level, domain-independent Geographic Display Agent (GDA) for use in DARPA's Strategic C3 Experiment. The GDA would assume total responsibility for the production of geographic-based displays, including the generation of background maps and the generation, placement, and symbol-collision resolution of user annotations. It would provide the application programmer with a high-level, object-oriented interface for constructing geographic-based displays, automating many of the functions a cartographer normally performs when creating geographically oriented displays. The GDA would reduce the graphics expertise required for developing aesthetic displays and would permit the programmer to focus on application issues, for which he is most qualified. The GDA's primitives would include maps, display and placement precedence control, and a general annotation mechanism, as well as traditional graphics primitives such as text, vectors, areas, lines, and swaths. Coordinates would be specified in latitude and longitude, with the GDA assuming responsibility for the transformation to screen coordinates for the particular map projection being displayed.

The remainder of this section describes in detail the functionality and architecture of the GDA.

4.4.1.1 GDA functionality

The GDA provides the C2 application programmer with an object-oriented interface that automates many of the functions a cartographer normally performs when creating geographically oriented displays. Its purpose is to allow the application programmer to concentrate on deciding which objects to display rather than the mechanics of actually generating an aesthetic display. The GDA achieves this by handling many of the important details surrounding the creation of a geographic display that would normally burden the application programmer. It handles issues such as ensuring that alphanumeric information does not overwrite and thus obscure other alphanumeric information; managing the display precedence of objects to avoid obscuring important display information; managing object placement order, so that those objects considered most important are placed closest to their desired positions; and transforming geographic coordinates into screen coordinates on a background map. The user can specify that a symbol with a specific label be positioned at a particular latitude and longitude, without having to perform the mathematics to calculate where it would be located on a Mercator Projection map. If there is already a symbol at that location, the GDA will relocate the one with the lowest placement precedence to an available screen location.

The user or his application program generates a text file or an otherwise sequential text stream that specifies the desired graphics display. This specification includes title,

classification, and date; a background map area; and a list of the desired graphics with their intended map coordinate locations. The GDA processes the textual specification, placing and drawing symbology on the requested background map according to various positioning criteria, and creates an aesthetic, human-readable display representation tailored to the connected display device.

A command to the GDA may include both explicit and implicit statements about a given object. The explicit statements are those that appear in the command stream; implicit statements result as a consequence of assigned defaults for various display parameters. These defaults have been selected to reflect the display properties generally desired for a given type of object. For example, the order in which an object is displayed is implicitly determined by the object type, e.g., marker, label, linear, or area object, and the order in which it appears in the command string. These and other defaults may be overridden, if desired, to accommodate unusual display objectives.

4.4.1.2 GDA objects

The GDA supports four basic kinds of graphic objects: markers, tags, linear objects, and areas.

Markers

A marker is an icon or text string used to denote the spatial location of a discrete item of interest, such as an aircraft, city, or reference point on a map. Markers are positioned at or as near as possible to their intended map locations, although placement conflicts with other GDA objects may result in markers being displaced from their intended locations.

Tags

A tag is an icon or text string used to annotate markers, linear objects, or areas. Tags are connected to the elements they annotate by a connecting line and are freely displaced from the referenced object, depending upon presentation requirements. Markers and tags may have the same visual presentation, that is, text may appear in markers, and graphics in tags. The principal difference between a marker and a tag lies in the default placement and display precedence values applied to them and their treatment in the display creation process.

Linear objects

Linear objects include lines and arcs. Linear objects are used to represent perimeters, paths, course tracks, or vectors. While they may distinguish regions of interest by defining the boundary of an area, they have different representation and display attributes than areas.

Areas

Areas include circles, sectors, and irregular polygons. They are used to represent two-dimensional regions of interest. They may be filled or unfilled, and they may be optionally bounded by a visible perimeter. The interior region of an area may be a factor in the display management, as opposed to merely the visible perimeter (as is the case with linear objects).

4.4.1.3 Object attributes

Display objects can have a variety of attributes that determine when, where, and how the object is to be displayed. These attributes include collision avoidance, placement precedence, display precedence, locations, and color.

Collision avoidance

The GDA includes an automatic collision-avoidance mechanism. This mechanism prevents certain object types from being written on top of and obscuring one another, thus reducing local clutter on the display and enhancing overall display intelligibility. As currently implemented, the GDA supports collision avoidance between textual object types such as tags and markers. While the user cannot disable the collision-avoidance mechanism, he can affect the results by ordering the objects being displayed or by manipulating the placement-precedence values assigned to various objects.

Placement precedence

The order in which an object is positioned on the screen is determined by the object's placement precedence. An object's placement precedence is represented as an integer value from 1 to 100, with a default GDA value associated with each object type. Objects of higher placement precedence are positioned before objects of lower precedence. In the event that multiple collision-avoidance objects are to be placed at the same screen location, the object with the highest placement precedence is placed first, at its indicated location. All other objects are displaced from the intended location according to their relative placement order. If two or more objects in the input stream have the same placement-precedence value, they are placed according to the order in which they are encountered. The user can explicitly set placement precedence as a parameter of an object, or can allow the system to assign a default value based on its object type.

Display precedence

The order in which objects are displayed on the screen is determined by the object's display precedence. Display precedence is represented by an integer value from 1 to 100, with an object of higher display precedence value visually overlaying a coincident object of lower display precedence. Each object, by virtue of its type, has a preassigned default display precedence based on normal expected use. Areas have been assigned the

lowest default display precedence and are drawn first. They are overlaid by linears, which have the next highest default display precedence, followed by tags, and then markers. The default display precedence for an object can be modified to create more atypical displays. For example, if a user wanted an area (e.g., "heavy weather") to obscure linears within that area, he would set the display precedence of the weather element to be higher than that of linears but lower than that of tags or markers.

Locations

All locations are specified in geodetic coordinates, either explicitly or via reference to other objects. The simplest way to specify a location is to supply the actual latitude and longitude values. Individual values are specified in degrees and fractions of a degree within a particular hemisphere, e.g., "N 35.5 W 4.25" corresponds to 35 degrees 30 minutes north latitude, 4 degrees 15 minutes west longitude.

A second way to specify a location is to refer to a previously defined location. Markers can be assigned a name, and that name can be used instead of geodetic coordinates to refer to a particular location (i.e., the location of the object). For example, if the coordinates "N 33.98 W 118.45" were associated with the marker named $ISI, then one could substitute the string $ISI for a coordinate element in a subsequent object to identify its intended location.

A third way to specify the location of an object is to give its location relative to an actual coordinate value or a previously defined location (as described above). The relative location is specified in terms of the angle (degrees) and distance (miles) from the indicated location. For example, 30 miles east of ISI could be expressed as either "A 90 D 30 N 33.98 E 118.45" or "A 90 D 30 $ISI".

Color

Color provides a valuable mechanism for discriminating between classes of objects being displayed, as well as visually reducing the apparent clutter of a particular display. The user may specify the desired color of an object to be displayed on the screen in lieu of accepting a default color. The GDA maps that specification into the nearest available color on the particular display device being used.

Color is expressed as three integers corresponding to intensity, hue, and saturation, similar to the approach adopted for the Tektronix 4027 color standard. The model is based on the location of a point within a color cylinder as described by the three integer values. Intensity is associated with the location of the point along the central axis of the cylinder; hue is determined by the angular distance of the point around the cylinder; saturation is determined by the radial distance of the point from the central axis of the cylinder. Intensity is the lightness of the object expressed as a percentage, with 0 being darkest and 100 being lightest. Hue is the characteristic color expressed as

an angle around the cylinder, with blue = 0, magenta = 60, red = 120, orange = 150, yellow = 180, green = 240, and cyan = 300. Saturation is the percentage of pure hue mixed with white of the same intensity, with 0 representing neutral and 100 representing pure color. If the user does not specify color for an object, the GDA supplies a default color of medium-intensity white.

Foreground overlays

The GDA supports one or more foreground overlays in addition to the background in which the map and most user-specified objects are placed. Unlike the background, in which newly drawn objects obscure previously drawn objects, foreground overlays allow objects to be drawn that do not obscure previously drawn objects. This is accomplished by using "transparent" colors for the foreground overlays, colors that change either or both the intensity and chromaticity of the background object. The overlays allow multiple types of information to be simultaneously visible on the screen with a minimal increase in display clutter.

The primary use of overlays is for defining areas or regions of interest rather than discrete locations, although there is nothing to prohibit the latter. Each foreground overlay provides one or more transparent colors, with collision avoidance and display precedence assigned to objects just as in the background. An object is normally drawn in the picture background. It can be placed in a foreground overlay by specifying the number of the overlay in which the object is to be drawn. Three overlays are available: numbers 1 through 3 are transparent.

4.4.1.4 GDA architecture

The three basic components of the GDA are the Graphics System, the Display Agent, and the Map Manager. They are described in detail below and shown in Figure 4-1.

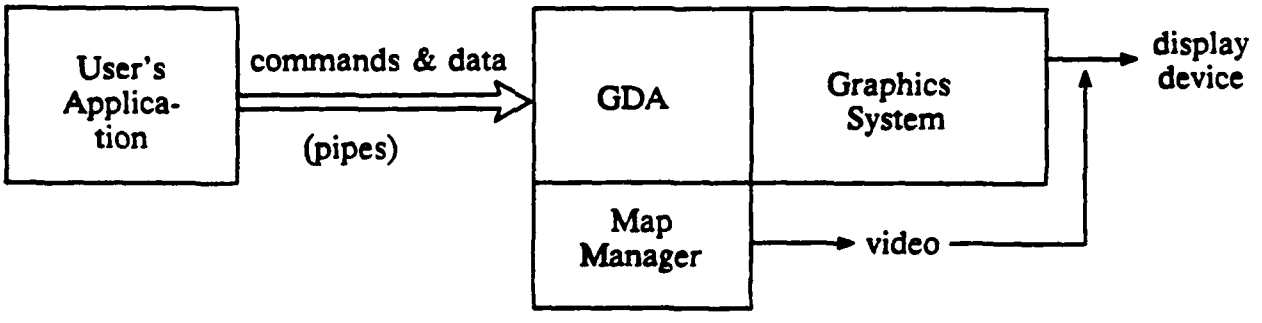


Figure 4-1: Geographic Display Agent architecture

Graphics System

The Graphics System was developed by ISI for DARPA for use in command and control. The system is both display-device independent and network distributable. The system's display-device independence permits graphic application programs to be written without regard to the particular type or location of the display device upon which the graphic output will ultimately be displayed. The system provides the application program with a set of generic, device-independent, two-dimensional graphics primitives by which pictures can be described and interacted with at the application program level. The system maps generic graphics primitives into the capabilities of the particular display device being used. A variety of application programming languages and output device types are supported.

The Graphics System has been designed to be network distributable across multiple host computers. Distributability permits the graphic display device and operator to be located away from the host computer on which the application program is run, possibly at a small, remote-site computer. Distributability allows the system to be flexibly configured to use available computation and communications resources, particularly important in crisis situations where resource requirements and availability cannot be predicted in advance. As currently implemented, the system can be distributed between PDP-10s, KL-20s, SMI SUNs, and VAXes connected by the ARPA-Internet, Packet Radio, or Wideband Packet Satellite. This system has been used in the past for Situation Display, an application-specific, natural-language-based naval database query and display system. It is now providing the graphics capability for the GDA.

Display Agent

The Display Agent code comprises the second major part of the system. It converts the user's textual geographic display description into display primitives for the Graphics System. Control operations initiate and terminate Graphics System operation, determine when picture data transmission is initiated and when it is complete, and request and retrieve information (such as error status) from the Display Agent. Data commands identify the background map to be used, objects to be displayed, and related graphic and overlay information such as circles, bands, lines, or arcs.

Map Manager

The Map Manager provides background maps in response to user requests. The Map Manager may dynamically construct the requested map from a stored map database, or it may incorporate stored map files or video images. The Map Manager produces maps of any part of the world on demand; they are then included in the background of the requested information display.

The maps may be produced in any of several ways. Custom maps can be computed at

run-time from a map coordinate database: they exactly match the user's map request (although these custom map images will be expanded vertically or horizontally to completely fill the graphics display area). Precomputed maps can be "stored" in graphics files containing map images, and then reproduced upon the display surface. A third type of map can be retrieved from a video-disk-based map database and mixed with the video signal from the graphics output device to create an underlying map background. All three types of maps may be used in the same session, with the GDA automatically selecting the appropriate source based upon the immediately preceding requests and the availability of suitable stored map images. Placement of data on the map images is adjusted automatically to correspond to the coordinate boundaries of the selected map image.

The Display Agent and the Graphics System are "linked" together and run as a separate process (in a separate address space) from the user's application. The user's application program communicates with the Display Agent code via the UNIX "pipe" mechanism, with the application program submitting control and data commands to the pipe for processing by the Display Agent.

4.4.2 Internet Concepts

4.4.2.1 Hierarchical naming system

The major progress in the Domain Naming System came in two areas: basic host name services and new services.

ISI provides the software used for the critical top levels of the name space. This software answers queries for the parts of the Internet (such as the MILNET) that do not provide their own name service and forwards queries for the hundreds of domains that have been delegated to the appropriate name servers. The aggregate query rate for the top-level servers has grown to several queries per second, and this is currently handled by four redundant servers. During the past year, we have debugged and demonstrated the ability to redeploy the hosts providing this top-level service, and have assisted many sites in setting up their own name spaces and servers under the top-level domains. We have improved the performance of the top-level service, provided logging and measurement tools, and added new capabilities to the top-level service to support new applications.

The domain name resolver for the Xerox 8010 Mesa development environment is operational, including well-known-service queries and pointer queries. This client implementation has been tested against both the TOPS-20 and the Berkeley UNIX servers.

This operational experience has provided confidence in the database delegation, caching, and timeout features that are the basis of the Domain Naming System. This is important in light of the explosive expansion of the Internet.

The new service with the most widespread consequences is the mail exchange, or MX, datatype. This new type of information was added to the Domain Naming System to decouple mail addresses from specific hosts. This allows mail to an organization to be independent of the status of a single host, and also allows organizations that are not directly connected to the Internet to have one or more mail-forwarding hosts represent them on the Internet as if they were directly connected. This feature is being used to allow CSNET, UUCP, and BITNET mail destinations to use normal mail addresses, and will introduce needed uniformity in mail addressing.

The Domain Naming System is also being used as a distributed database tool by other researchers. Examples include its use by several sites for direct user (as opposed to host) registration for mail and other purposes. The services provided by the Domain Naming System have been selected for use as the foundation for the NETBIOS name service. These experiments will lead to new functionality in the Domain Naming System and will also serve as a useful stimulus for refinement of the Domain Naming System mechanism. We have provided guidance and support for many of these efforts.

4.4.2.2 Computer mail interoperation

The evolution of large electronic mail systems testifies to the increasing importance of electronic mail as a means of communication and coordination throughout the scientific research community. These systems include the DARPA Internet mail system (ARPA-Mail), the GTE Telemail system, the MCI Mail system, and the IEEE Compmail system (a Dialcom system). Until recently, these systems have operated autonomously, and no convenient mechanism has existed to allow users of one system to send electronic mail to users on another system. Intermail is an experimental mail forwarding system that allows users to send electronic mail across such mail system boundaries. Users on each system are able to use their usual mail programs to prepare, send, and receive messages. No modifications to any of the mail programs on any of the systems are required. [13]

Intermail has a login account and a mailbox on each of the commercial mail systems it services. A user on a commercial system sends mail destined for a user in the ARPA-Mail world to the "Intermail" mailbox on his local commercial system. The Intermail program periodically picks up this mail, determines the destination ARPA-Mail address, and turns the mail over to the Internet mailer for delivery to the appropriate Internet host. In the other direction, a user on an Internet host sends mail destined for a user on a commercial mail system to the ARPA-Mail mailbox of Intermail (i.e., "INTERMAIL@CISI.EDU"). Intermail periodically picks up messages from its

mailbox, determines the destination mail system, and sends the message using Telemailer, Compmailer, or MCImailer. The Intermail program examines the forwarding information contained in each message to determine the destination system and mailbox.

The Intermail experiment continues to provide regular service. We have implemented automatic program-generated error messages for the Intermail mail forwarding system. Error-processing capabilities were added to the system to enable it to return error messages generated by a remote ARPANET mailer to the original sender of a message, on a commercial mail system.

All of the mail systems involved have changed during the year (added features, etc.). This has required modifications to the Intermail programs to maintain compatibility. This year we added service for two additional Dialcom systems for NSF and ONR.

4.4.2.3 Studies, surveys, and specifications

A routing procedure called Cartesian Routing was developed that routes based on distance. Each packet carries the coordinates of the destination, and each node routes the packet to reduce the distance remaining to the destination. The Cartesian Routing procedure has the following properties:

- It scales to very large network size.
- It does not require large tables or routing information exchanges.
- It allows mobile hosts.

A simulator was constructed to test the performance of Cartesian routing on an existing network topology. Both single-level and two-level hierarchic routing algorithms were tested over all IMP source/destination pairs in the ARPANET. The results were compared to the Shortest-Path First (SPF) algorithm used in the Internet, which represents the theoretical optimum. Single-level Cartesian routing showed an average path length 27 percent above SPF, while the two-level hierarchy showed 17 percent greater path length than SPF. Two reports were written on a new routing mechanism for packet-switched networks, which has several advantages over currently used algorithms. The new routing mechanism has the ability to scale upward indefinitely, and it involves a relatively transparent treatment for mobile hosts, which are treated no differently from non-mobile hosts as far as intermediate routing nodes are concerned. [8,9]

The ISI Internet Protocols software was implemented on the Xerox Dandelion workstation, under XDE version 4.0. An enhanced version of the Xerox Ethernet SpyTool has been developed and is being used for performance analysis of TCP-based FTP over the Wideband Network. This SpyTool will also be used to analyze the

performance of new protocols such as the NETBLT protocol, which is currently being developed at MIT.

In the operation and the evolution of the Internet system, it is sometimes useful to survey or measure the implementation status and performance of particular protocols. As the Internet system evolves, and as flaws and ambiguities are discovered, specification documents are upgraded.

4.4.3 Multimedia Conferencing

Multimedia mail

The development of multimedia mail was one of the precursors of the work on multimedia conferencing. During the first part of the year, ISI completed a successful multimedia mail system (MMH) [14], which interoperated with other such mail programs implemented by BBN and SRI in the overall DARPA research program. The MMH multimedia mail user interface program, written in LISP for the Xerox 1108, was completed and distributed to interested users at other sites.

MMH allows the user to create a tree-structured message with the leaves of the tree being media elements of type text, bitmap, or voice. The message may be edited to rearrange the structure, and the individual data elements may be modified. MMH also allows the user to read messages from the mailbox, examine their structure, and display individual elements of messages.

A key concern in our development of MMH is presentation control. MMH allows the message composer to use spatial control for positioning message elements on the screen. When multi-element messages are received from other systems without explicit spatial information, MMH will use default positioning information. The default information uses history data to make an intelligent decision as to where to place message elements on the screen. A similar strategy was implemented for message composition and has been found to be useful in that context as well.

Subsequent to the completion of MMH, the Diamond multimedia mail system developed by BBN was converted to a new document format that is incompatible with MMH and the SRI system. This change was motivated in part by the need to transition to emerging international standards (ISO X.400) for multimedia mail. Given the current environment, we judged it best for our work on a multimedia conferencing system to be based on the Diamond system.

Image scanner

To allow the inclusion of offline material into multimedia mail and multimedia conference presentations, a scanner interface has been developed. We have programmed the IBM-PC to scan a document using an attached Microtek MS-200 scanner. The scanned data is stored as a standard bitmap file (RFC 797), which is then uploaded into a host. Hardcopy output may be obtained from an Imagen printer. The scanner program can also produce compressed files in the CCITT Group 3 format.

A Pascal program was developed to expand a standard bitmap file linearly by one-half so that the image scanned at 200 dots per inch will fill out an entire 8.5 x 11 inch page when output on the Imagen printer at 300 dots per inch. Another program can display a bitmap file on the Micro Display Systems' Genius 738 x 1004 bitmap display. Because the video display memory is only 128 Kbytes, it can display only about one-quarter of the bitmap file (the bitmap file is 466 Kbytes). A third program allows scanning a document, writing it to a file, reading from the file, clipping, writing the clipped data into a file, and displaying the clipped data.

4.4.3.1 Real-time conferencing

A real-time multimedia teleconferencing system has been constructed from components developed by forerunner projects. Packet voice and packet video facilities developed under the Wideband Communication project at ISI provide isochronous communication, while the MMCONF software developed by the Diamond project at BBN provides a shared workspace for text, graphics, bitmaps, and other media on a SUN workstation screen. The media components are being integrated together for easy user control of a conference, while individually they are being refined for improved performance.

In addition to our own project-specific experiments with conferencing, we have also used the system for meetings that discuss topics other than conferencing. We obtain valuable feedback from these non-involved users on how well the system works, and the users are able to meet without travelling. We look forward to providing this service on a regular basis in the future.

The first such use of the multimedia teleconferencing system was on April 1, 1986, between ISI and BBN. Participants included personnel from BBN, ISI, and SRI, plus sponsors from DARPA and NOSC. The teleconference was the culmination of several efforts:

1. Our packet video installation at Lincoln Laboratories was moved to BBN for ready access by the multimedia conferencing researchers there.
2. Performance of the Voice Funnel and Packet Video software was tuned to allow maximum throughput and to coordinate the simultaneous use of packet voice and packet video.

3. The Wideband Network stream service and QPSK modulation were made available in the new BSAT IMPs to provide the high bandwidth and low delay required for good packet video.

4.4.3.2 Packet video

During the first part of this reporting year, we continued development of the packet video system as part of the Wideband Communication project. A quantum improvement in performance was achieved with the implementation of a "variable-frame-rate" compression and transmission scheme. This scheme incorporates interframe coding to avoid processing and transmission of any parts of the image that are unchanged from the previous image. This allows a reduction in the data rate by as much as a factor of eight, or conversely an increase in the image update rate by the same factor. The system currently implements the latter technique, producing a variable frame rate (update rate) but a constant data rate.

Unfortunately, hurricane Gloria forced the cancellation of a planned demonstration of the packet video system. However, a full-duplex video connection was established between ISI and Lincoln Laboratories, with a simultaneous voice call also placed across the Wideband Network. This test was the flawless final performance of the PSAT IMPs before they were replaced with BSATs on October 1, 1985, as part of the Wideband Network upgrade.

Subsequently, the packet video system was moved to BBN for use in multimedia conferencing. Two-way simultaneous video connections are now made on a fairly routine basis. We have continued to refine the quality of the video based on our conference experiments. Further work would be required to install additional sites and to extend the data-transmission software for multipoint communication. More detail is provided in the following sections.

Video System overview

The Video System design can be outlined as follows. There are three sections: a front-end video I/O system (a commercial graphics system), an image-processing section (composed of a processor from the commercial system and a special processor built by ISI), and a data transmission section to connect to the network. The video I/O system consists of a camera, a display monitor, and an image data buffer for images scanned in from the camera or for display. The image-processing section includes the system controller, which manages all three sections, and the data-compression processor. Pictures are input from the camera, compressed, and sent to the network node for transmission. Incoming data is received from the network node, expanded, and displayed. The data transmission section includes a high-speed data link that exchanges data between a compressed-data buffer, used by the image-processing section, and a message protocol program that formats the data for the network node.

Inter-frame and intra-frame compression algorithm

Most video-compression systems use a frame-differencing algorithm to detect motion or the lack of motion. Appreciable motion generally requires a compression system to operate at its maximum update rate, perhaps at the expense of image quality. Less motion may allow lower data rates or improved image quality. Lower data rates can be achieved by some combination of lower frame update rates or less data sent per frame.

The scheme implemented in the ISI system works somewhat differently. Primarily due to limitations in processing power, the image quality is not variable. When there is appreciable motion, more time is required to process all the changing parts of the image, so the image update rate is reduced. When there is little motion, it is possible to increase the image update rate for maximum motion fidelity, or to reduce the data rate to minimize network loading. The system currently implements the former technique, producing a variable frame rate (update rate) but a constant data rate.

The image-processing section employs a two-phase data-compression scheme and a single-phase data-expansion scheme. The scanned image is subdivided into 16 x 16-pixel blocks, then all processing is done blockwise. Data compression performs inter-frame differencing followed by intra-frame compression.

To construct a list of the blocks to be updated, the inter-frame differencing algorithm compares each block in the new frame with the corresponding block in the previously processed frame. To determine whether the new block is sufficiently changed to require updating, values of selected pixels in the new and previous blocks are compared. The update decision depends on whether the number of pixels that crossed a difference threshold exceeds a certain count threshold. Blocks not selected for updating are ignored except for periodic refreshment.

Intra-frame compression is then performed only on those blocks of the frame identified for update: the image data is transformed blockwise by the Discrete Cosine Transform to produce frequency coefficients that can be systematically compressed. Since the blocks processed and transmitted comprise only a partial list of all 240 blocks in a frame, information about the location of the blocks in the frame must be sent along with the compressed data.

To deal with the mismatch between frame update rates for the received and transmitted pictures, the processing of each image frame has been broken into fixed 32-block units. The Discrete Cosine Transform processor is able to process 32 blocks of forward and inverse transforms in 1/30 second, the maximum frame rate from the camera. For each new frame to be transmitted, the changed-block list is divided into 32-block sets, with the remainder filled in by refresh blocks selected in a cyclic fashion

from the 240 blocks of a complete image. Each set is handled by the forward processing portion of a new unit processing cycle and then transmitted until the frame has been completed.

Received blocks are collected into 32-block sets, which are processed in the inverse portion of each unit cycle. Data expansion involves only intra-frame processing, which transforms compressed frequency-domain data back to image data using the inverse Discrete Cosine Transform. The resulting image blocks are inserted into the picture being received according to the block addresses received with them, but update of the image display is held until all the updated blocks of each frame have been processed.

Each unit cycle thus handles 32 blocks of forward and 32 blocks of inverse processing. By interleaving the 32-block units of forward and inverse processing, constant transmit and receive data rates are sustained while accommodating varying amounts of processing per frame. The data rate is fixed because unit processing cycles occur at a fixed rate and each cycle processes exactly the same amount of data. The frame update rate, however, can vary from 30 frames per second (keeping up with the full camera scan rate) when fewer than 32 blocks change per frame, to 3.75 frames per second when 8 cycles are required to cover the 240 blocks of a full frame. The 32-block interleaving allows the transmit and receive frame update rates to vary independently. End-to-end delay is also reduced, because time granularity is a fraction of a frame time instead of a full frame time.

The current image-processing scheme can be described as a "variable-frame-rate" scheme that operates at a constant data rate but with per-frame data compression that varies from 16:1 to 128:1. Starting from the 512 x 480 camera image, the cumulative compression is 16:1 (4:1 from subsampling and 4:1 from intra-frame compression to 2 bits per pixel). Compression from inter-frame differencing contributes the variable factor of 1:1 to 8:1. The data rate is constant, with a maximum setting of approximately 500 Kbps, but it may be throttled back arbitrarily with a corresponding reduction in frame rate. Current operation on the Wideband Network is limited to three-fourths of the maximum rate to allow for other simultaneous traffic.

Network protocols and software

The data transmission section of the Video System is responsible for the two-way exchange of data between the compressed data buffer and the Voice Funnel, which serves as a network gateway. The Voice Funnel program runs on several processors of a BBN multiprocessor Butterfly computer.

The physical pathway between the video-compression hardware and the Butterfly is a 2 Mbps HDLC serial data link. Built into the video hardware is a data-link controller that transmits and receives packets over the link. At the Butterfly end of the link is

the Packet Video Program (PVP) that manages packet exchange with the data link controller.

PVP has two I/O ports, one for the data link and another to connect to the Voice Funnel. On the transmit path, PVP completes the packetization of video data according to the Packet Voice Protocol, performs packet bookkeeping chores, and regulates the flow of packets to the Voice Funnel. On the receive path, PVP groups and reorders packets to ensure that all blocks processed in one 32-block cycle belong to a single frame. PVP does further consistency checks and packet bookkeeping, and regulates the flow of packets to the video hardware. PVP functions asynchronously with the Voice Funnel but synchronizes packet flow with the video hardware, on a cycle-by-cycle basis, by exchanging control packets with the video control processor. PVP also provides an operator interface for establishing network connections and provides statistical displays for operational monitoring. PVP serves as a network host, exchanging outgoing and incoming communication packets with the network gateway, the Voice Funnel.

A substantial performance improvement has been achieved by implementing the Stream protocol (ST) in PVP to replace the IP datagram transmission used initially. On the Wideband Network, the ST protocol allows the use of reserved stream bandwidth that cuts transmission delay in half and minimizes delay variations. In addition to supplying the data packet headers for ST, PVP is responsible for the control packet exchanges necessary to establish and terminate the ST connection. In the current implementation, no NVP-style IP packet exchange precedes the opening of the ST connection as was done in the Packet Voice system.

Packet Voice

The Packet Voice facility is part of the multimedia teleconference system, but it can also be used for normal point-to-point voice calls. We want the Packet Voice facility to be available for general use and reliable enough to invite such use.

Reliability and availability have been improved by the implementation of a second-generation Packet Voice system. The first-generation system consisted of Packet Voice Terminals (PVTs) developed by Lincoln Laboratories with inserted Switched Telephone Network Interface (STNI) cards developed by ISI to provide access from standard telephones. For the second-generation system, BBN has adapted the ISI STNI design to fit in a Multibus cardcage incorporated into the Voice Funnel machines. The BBN STNIs use a high-speed serial interface, as opposed to the parallel interface used in the PVTs; we wrote a new version of the STNI software to support the serial interface on the BBN STNI. A program that performs the equivalent of four Lincoln PVTs runs in the same Butterfly computer as the Voice Funnel program.

Implementation of the Packet Voice system in the Butterfly increases the available number of STNIs and makes use of the greater capacity of the Butterfly to increase the functionality of the PVT program as compared to the Lincoln PVTs. Reliability has been improved by replacing the aging PVTs with the Butterfly VT implemention, and overall reliability of the packet voice facility has been improved substantially due to the improvment in general Wideband Network reliability since it was upgraded (see Section 4.4.4).

When the Packet Voice system is used for teleconferencing, packet communication is the easy part. One of the most difficult aspects of meeting-style teleconferencing is the voice input and output. It is very hard to have an interface that is both comfortable and natural for the participants while it avoids acoustic echo feedback in the room.

We have conducted several multimedia teleconferences using various combinations of microphone and speaker equipment; input sensitivities, output levels, and turnaround timing in echo suppression; and 2-wire vs. 4-wire interfaces. This is still an area of exploration.

4.4.4 Supercomputer Workstation Communication

The primary goal of the Supercomputer Workstation Communication project is the development of appropriate interaction protocols between workstations and supercomputers. The experimental environment includes the Wideband Network. Thus, a portion of the effort of this project is directed to testing the performance of the Wideband Network and tuning standard protocol implementations to achieve good performance when using it.

4.4.4.1 Wideband Network upgrade

During this year the Wideband Network underwent a major system upgrade with significant changes in many components. An early task of this project was to establish performance baselines for the new Wideband Network.

The upgrade affected the basic elements of the network node, which consists of an Earth Station (antenna and RF converters), ESI (Earth Station Interface), and Satellite IMP (Interface Message Processor). The upgrade included the following:

- installation of larger seven-meter antennas and more reliable RF converters
- new ESI-Bs to replace the ESI-As
- new BSATs to replace the PSAT network node IMPs

Replacement of the remaining five-meter antennas with seven-meter antennas provides stronger received signals. The old IMPs, called PSATs (Pluribus Satellite IMPs), have

been replaced by new IMPs called BSATs (Butterfly Satellite IMPs), which use BBN's multiprocessor Butterfly computer. New versions of the Linkabit ESI with increased throughput and a new interface to match the Butterfly have also been installed. The changeover from PSATs to BSATs caused a major disruption for us as users of the network. However, reliability and performance of the network have improved substantially since the changes were completed.

One method by which we evaluate the reliability of the Wideband Network is to automatically run a test every hour to determine which sites on the network are accessible. Monthly reports are generated to summarize the test results; these reports serve as the primary status measure for the Wideband Network. To help differentiate between BSAT downtime and host downtime, the hourly tests have been augmented to probe BSAT internal ECHO hosts as well as real hosts. The results are reported separately for the two categories of hosts.

4.4.4.2 Performance of standard protocols

We are testing the performance of the standard IP/TCP-based Telnet and FTP protocols over the Wideband Network through Ethernets and gateways to demonstrate the utility of the Wideband Network for general-purpose data traffic. Preliminary tests between a VAX computer at BBN and a SUN workstation at ISI revealed several problems.

Hosts and gateways do not normally route traffic across the Wideband Network, but use the ARPANET instead. We devised methods to force the desired routing. With standard parameter settings, IP/TCP-based file transfer provides no more throughput on the Wideband Network than it does on the ARPANET (about 10 Kbps). We developed an enhanced version of the Xerox Ethernet SpyTool program to examine packet flow on the local Ethernet, and we then analyzed that data to find the cause of the low throughput. Timeout values in TCP are not a source of difficulty, but the "window" size must be as big as the implementations will allow to accommodate the unusually high number of bits in flight across the satellite network (the observed round-trip time was 1.8 seconds). Currently the window size is limited by TCP implementations; using a window size of 15360 bytes resulted in a maximum transfer rate of only 50 Kbps. However, the maximum window size allowed by the TCP protocol is still insufficient to achieve the maximum possible throughput on the Wideband Network. Therefore, alternative protocols will be required for the Supercomputer Workstation environment.

4.4.4.3 NETBLT protocol

One promising alternative to TCP is the new NETBLT protocol developed by MIT. We are participating with MIT in the testing of this protocol over the Wideband Network, analyzing observed behavior to find bottlenecks that must be eliminated. We have achieved a maximum transfer rate of 225 Kbps using MIT's IBM-PC/AT implementation of the NETBLT protocol. We observed several operational problems, the most serious being the limited rate at which the Ethernet interface in the IBM-PC/AT was able to receive packets. When packets are transmitted at a high rate on the Wideband Network, they tend to be aggregated together for transmission in a single burst. Since the gateways between the Wideband Network and local Ethernets have high throughput and low delay, packets received in a single burst are transmitted in rapid succession on the Ethernet. The spacing between packets may therefore be less than the minimum the IBM-PC/AT and its Ethernet interface can accommodate.

One solution to this problem is to require that the gateways space out the packets. Unfortunately, this limits throughput for more powerful hosts. We will solve the problem by implementing NETBLT in a SUN workstation to use in place of the IBM-PC/AT. Then we will further refine the transfer parameters for NETBLT to determine the optimal values to match the performance characteristics of the Wideband Network.

The current NETBLT protocol is based on the lower level IP protocol, as is TCP. As part of the next phase of the SWC project, we may modify the NETBLT protocol to be used with the Stream protocol (ST) instead. We participated in the development of the ST protocol for the transmission of packet voice and video over the Wideband Network. The bandwidth-reservation features of ST mesh well with the requirements of NETBLT, and would allow higher throughput with lower delay through the Wideband Network.

4.4.4.4 Network monitoring tools

To monitor the performance of the TCP and NETBLT protocols in Wideband Network testing, we have used a Xerox 8010 XDE workstation running the SpyTool program to monitor the packets as they cross the Ethernet. This program has proven valuable for recording the sequence and timing of the packet stream. When testing the TCP protocol, we observed that the sender encountered a flow-control window of zero size, leading to the conclusion that the limiting factor is the size of the receive buffer.

Early use of the SpyTool was limited by the discovery that low-level communication software in the XDE was unable to receive odd byte-length packets or packets longer than 576 bytes. We isolated these problems and assisted Xerox in the installation of bug fixes. We have done additional work on the SpyTool program to increase the maximum packet rate it can monitor.

4.4.4.5 Supercomputer survey

We interviewed several scientists who use supercomputers, in order to learn what facilities currently exist and what new ones are desired. These interviews will give us more insight into the communication functions required. We are participating in the DARPA Internet Scientific Networking Task Force to explore these issues with researchers from other institutions.

4.5 IMPACT

The Command and Control Graphics project has demonstrated an important new approach to structuring the components of command graphics systems. The Geographic Display Agent provides improved ability to portray command data in a geographic context. The impact of packet-switched computer communication has been enormous. The success of the Defense Data Network (DDN) is one example of the importance of this work. Further enhancements of such systems depend on continuing research in computer network technology. The work reported here extends the previous capabilities. The Multimedia Conferencing effort explores new capabilities for user-to-user information exchange. This effort has demonstrated that packet video communication is possible in the Internet environment. Further development of an integrated multimedia conferencing system will enhance our capability for collaborative work in this environment. The Domain Naming System extends the power of the Internet system to identify resources by name, because it increases the flexibility and dynamics of name management. The computer mail interoperation effort extends the communication capability for this popular mode of computer-based communication. The expanding use of the Internet system requires a continuing effort to provide up-to-date knowledge of system design, specifications, and performance. In the Supercomputer Workstation Communication project, the development of high-capacity communication systems will provide wider accessibility to supercomputer facilities so that the power of supercomputing can be used more efficiently.

4.5.1 Command and Control Graphics

The principal impact of this work will be felt in military environments where command mobility is paramount and a portable, network-based graphics capability is important for information presentation for the command decision process. Completion of a conversion effort to the SMI SUN has made the Graphics System available in mobile environments. Development of the Geographic Display Agent has facilitated development of decision aids and other application software that has a strong, geographically oriented information presentation requirement. The work also serves as a model for future tool development by clearly demonstrating the utility of developing flexible and adaptable tools that are not tied to specific hardware and that incorporate "intelligence" about particular areas of functionality to be supported.

4.5.2 Internet Concepts

4.5.2.1 Hierarchical naming system

The Domain Naming System has replaced the previous HOSTS.TXT method on a large, and growing, number of hosts in the Internet. The operational feasibility of the scheme has been demonstrated, and the combination of heterogeneous management and hosts, datagram access, and size makes it a unique distributed database. We are cooperating in transition planning for the MILNET, several steps toward full conversion have been made, and conversion assistance aids are in place or under development.

The Domain Naming System has been used to enhance the capabilities of the existing mail system through the MX scheme for mail repositories and routing. The system is forming the basis of a name service for NETBIOS, and several sites are actively using the extensibility features to provide distributed databases for new distributed applications.

4.5.2.2 Computer mail interoperation

The experiment in computer mail interoperation has proven successful with the test community. The capability to extend this popular service beyond the boundaries of the Internet may have significant impact on the options available to provide government contractors access to common computer mail facilities. Many contractors have been granted access to the ARPA-Internet simply to enable them to communicate with the government. It may now be feasible to direct some of these contractors to use commercial computer mail systems and the Intermail interconnection service.

4.5.2.3 Studies, surveys, and specifications

The selection of the IP and TCP protocols by the DoD as the basis for a DoD internetwork protocol standard shows the impact of the work of the DARPA community on DoD communication systems. The development of the Defense Data Network (DDN) by the DCA is a major use of these protocols in an operational military system. This influence is demonstrated further by mounting requests for information about IP/TCP from companies interested in producing commercial products or bidding on government contracts.

Through our participation in the Internet Working Group meetings and in technical meetings with other contractors, we have successfully influenced the development of many protocols and protocol features. Our publication in conferences, journals, and newsletters extends the impact of this work to others who design similar systems.

4.5.3 Multimedia Conferencing

Computer mail was the most significant new use of the communication capability provided by packet-switching networks. Expanding from text-only mail to multimedia mail was a quantum-level improvement. The ability to communicate diagrams or maps and to then talk about them through multimedia conferencing will tremendously increase the effectiveness of remote communication. The combination of text, speech, graphics, and facsimile into a common framework and data structure may have substantial impact on other applications as well.

We have taken the lead in developing the video and image media for real-time conferencing, and we coordinate the overall effort. The requirements of packet voice and video have been the primary drivers in the development of the high-bandwidth packet networking technology represented by the Wideband Network.

The U.S. military has a pervasive need for timely, reliable, and effective information transfer, both in times of crisis and as a part of normal operations. The purpose of multimedia conferencing is to enhance the productivity of communications between individuals through the use of computers and networking.

4.5.4 Supercomputer Workstation Communication

This project has significant potential impact in three areas. First, high-bandwidth access to supercomputers at a limited number of locations will be available to researchers in DoD-sponsored programs at a larger number of remote locations. Second, the protocols and routing methods developed to provide supercomputer access will also allow the Wideband Network to be used for some of the traffic now flowing over more heavily loaded terrestrial networks in the DDN. Third, new techniques and applications of high-capacity computer communications technology developed by this project can be used in military systems to be deployed in the future.

4.6 FUTURE WORK

We will continue to work in three areas: Internet Concepts, Multimedia Conferencing, and Supercomputer Workstation Communication. The Command and Control Graphics project has been completed.

4.6.1 Internet Concepts

4.6.1.1 Hierarichical naming system

We have finished development of the basic service support, and anticipate only a low level of maintenance activity. We will also complete documentation of the protocol and transition aids for the Internet community. We will continue to provide guidance and support for researchers that seek to use the Domain Naming System as a basic tool, either to provide new types of information or as a building block for higher level services. An important part of this service involves coordinating new features and datatypes to insure coherent growth and to avoid inconsistencies, duplicate facilities, and additions that limit the scope of use.

4.6.1.2 Computer mail interoperation

This effort will continue to operate the Intermail program and will help to develop the Commercial Mail system (see Chapter 10), a fully supported service to replace our experimental service.

4.6.1.3 Studies, surveys, and specifications

This effort will continue to conduct surveys and specifications of Internet protocol implementation features, conduct numerous performance measurements, and report the results to the ARPA-Internet community. In addition, we will produce up-to-date documents of the protocols used in the ARPA-Internet community as needed, and manage the assignment of protocol parameters to network experimenters as needed.

4.6.2 Multimedia Conferencing

We will continue to develop and refine the prototype conferencing system. The media components will be integrated together for easy user control of a conference, and refined individually for improved performance. We look forward to providing the teleconference meeting service on a regular basis, in order to obtain valuable feedback on its performance.

The user interface for the packet teleconference system will be refined so that the system can be operated by end-users with no knowledge of the Video System, the Wideband Network, or other internals. The teleconference system should be available on a moment's notice, and should not require expert supervision.

Deployment of the packet teleconference system will be expanded from the existing sites at ISI (Marina del Rey, California) and BBN (Cambridge, Massachusetts) to include new sites at the DARPA offices in Washington, D.C., and at SRI International in Menlo Park, California. This effort includes the installation of additional hardware, plus the development of multidestination support in the Packet Video protocol implementation. This will permit the image seen by one camera to be distributed

through the broadcast medium of the satellite channel to all sites participating in the conference.

A new emphasis in future multimedia conferencing work will be the development of a standard architecture and protocols for conferencing. The purpose is to create standard building blocks and models for the conferencing infrastructure. This will avoid duplication of effort between media or contractors, and will promote an open architecture conducive to experimentation. This development will be organized into three levels of abstraction: basic services, control framework, and management procedures.

4.6.3 Supercomputer Workstation Communication

No supercomputers are currently accessible from the Wideband Network through high-bandwidth paths. However, we expect such access to become available soon. We will test the IP/TCP-based protocols in the supercomputers; we expect that tuning of the implementations will be required to achive desired performance levels, as we found in similar tests on smaller computers.

We will also work on the second step of our program, attempting to move past the traditional remote access services provided by these protocols and exploring more sophisticated forms of interaction between workstations and supercomputers. We will work on the development of the "network virtual window package" that is designed to allow the computation site to call window package routines that communicate over the network and to execute those window package functions locally on the workstation.

The third step of our effort is to approach interprogram communication at a much higher level of abstraction. We will explore the feasibility of configuring multimachine cooperating computations by connecting the outputs of one program to the inputs of another program. There are many obstacles to putting this simple notion into practice: format differences, timing dependencies, different parameter-passing methods, and differing levels of data abstraction. We will attempt to develop methods to overcome these obstacles.

References

1. Postel, J., and J. K. Reynolds, "File Transfer Protocol (FTP)," RFC-959, USC/Information Sciences Institute, October 1985.
2. Reynolds, J. K., and J. Postel, "Assigned Numbers," RFC-960, USC/Information Sciences Institute, December 1985.
3. Reynolds, J. K., and J. Postel, "Official ARPA-Internet Protocols," RFC-961, USC/Information Sciences Institute, December 1985.
4. DeSchon, A., "A Survey of Data Representation Standards," RFC-971, USC/Information Sciences Institute, January 1986.

5. Mockapetris, P., "Domain System Changes and Observations," RFC-973, USC/Information Sciences Institute, January 1986.

6. Reynolds, J. K., R. Gillman, A. Brackenridge, W. Witkowski, and J. Postel, "Voice File Interchange Protocol (VFIP)," RFC-978, January 1986.

7. Jacobsen, O., and J. Postel, "Protocol Document Order Information," RFC-980, March 1986.

8. Finn, G., "Routing and Addressing Problems in Large Metropolitan-scale Internetworks," SRNTN 44, USC/Information Sciences Institute, July 1986.

9. Finn, G., "The Application of Cartesian Routing to Networks of Irregular and Unstable Topology," SRNTN 45, USC/Information Sciences Institute, July 1986.

10. Reynolds, J. K., "ISI-HOSTS," ISI Internal Memo, USC/Information Sciences Institute, November 1985.

11. Reynolds, J. K., and J. Postel, "Instructions for the Facsimile Machine," ISI Internal Memo, USC/Information Sciences Institute, November 1985.

12. Postel, J., "Protocol Document Order Information," ISI Internal Memo, USC/Information Sciences Institute, January 1986.

13. DeSchon, A., *Intermail, an Experimental Mail Forwarding System*, USC/Information Sciences Institute, RR-85-158, September 1985.

14. Reynolds, J. K., J. Postel, A. Katz, G. Finn, and A. DeSchon, "The DARPA experimental multimedia mail system," *IEEE Computer*, 18, (10), October 1985. Also published as USC/Information Sciences Institute, RS-85-164, December 1985.

15. Bisbey, R., and D. Hollingworth, *A Distributable, Display-Device-Independent Vector Graphics System for Command and Control*, USC/Information Sciences Institute, RR-80-87, July 1980.

16. Beebe, A., and S. Casner, "ISI Packet Video System," ISI Internal Memo, May 1986.

5. ADVANCED VLSI

Research Staff:
George Lewicki
Ron Ayres
Jeff Deifik
Joel Goldberg
Wes Hansford
Lee Richardson
Craig Rogers
Carl Service
Bing Sheu
Barden Smith
Jeff Sondeen
Vance Tyree

Research Assistants:
David Hollenberg
Ming Hsu
Wen-Jay Hsu
Shih-Lien Lu
Mahesh Patil
Vijay Sharma
Je-Hurn Shieh
Eric Shih

Support Staff:
Barbara Brockschmidt
Mike Curry
Sam Delatorre
Terry Dosek
Kathie Fry
Terri Lewis
Christine Tomovich

5.1 PROBLEMS BEING SOLVED

The foundation of modern military weapons systems is electronics. As levels of integration on silicon climb higher and higher, large, complex systems are going to fit on just a handful of chips. Restricting future military systems to the use of standard parts is analogous to insisting that all future movies be composed from ten prefilmed standard scenes.

Military electronics will have to enter the world of custom and semi custom integrated circuits. Military system architects must learn to design directly on silicon, taking advantage of all the opportunities afforded by that medium and at the same time learning to avoid all the pitfalls.

A large fraction of DoD's systems contractors do not own their integrated circuit fabrication facilities. These contractors are having a very hard time obtaining custom and semi-custom fabrication for military weapons systems. The reasons for this are manyfold:

- The DoD typically needs small volumes. In fact, the DoD represents only some 5 percent of the industry's total market. The economics of the industry dictate the selling of wafers in very large volumes. The only reason a fabricator will agree to produce small volumes is in the hope of getting a customer to high-volume production.
- The DoD must buy against stringent military specifications (MIL-SPEC), which vendors must certify as having been met. The majority of the industry's customers do not require such certification. The result is that DoD does not access a large number of fabricators who have high-quality processes but who do not want to bother with the business of line and parts certification.

- The interface to the industry is complex. Each vendor has its own proprietary design rules, obtainable only after the signature of nondisclosure agreements. Each vendor has its own proprietary design libraries, obtainable only on the condition that designs based on these libraries be fabricated only by the vendor owning the libraries. Vendors have proprietary design systems with similar restrictive conditions applied to their usage.

  The result is that a chip designer is forced to develop a separate interface to each fabricator. There is an extremely high probability that chips designed according to a set of proprietary design rules cannot be fabricated by a multiple vendor base. This means that the production of the chip cannot be put out for bid. It means that if the single vendor supporting those design rules decides to move on to another technology, the design can no longer be fabricated and all the time invested in the design will be lost. It means that if the vendor temporarily loses his process, fabrication of the chip and then development and production of the system will be delayed.

- The cost of doing an application-specific design is very high. It typically means associating oneself with one vendor. It means paying on the order of $30K to $50K for each iteration of a prototype design. This is the amount necessary to generate masks and to buy a minimum lot of ten wafers, each of which holds from one to several hundred copies of the design. The number of copies needed, however, is not several thousand; it is nearer to several tens. Thousands of circuits are generated because that is the minimum number that can be bought.

Before the DoD can expect to benefit from the use of application-specific VLSI in its weapon systems, it must provide its systems contractor community with a much more effective interface to the U.S. semiconductor industry than the one that presently exists. Such an interface is possible, and MOSIS is developing it by defining a standard interface through which many fabricators can be accessed. This interface includes a set of non-proprietary design rules applicable to a multiple vendor base and scalable with the decreases in feature size expected over the next several years. In addition, MOSIS is developing a multiple vendor base within the semiconductor industry capable of supporting that interface. By aggregating the integrated circuit needs of a whole community and representing them as one to that vendor base, MOSIS is benefiting both the vendors and the designers. By using the MOSIS Service, a large number of designers are able to develop their designs with little or no investment of the fabricators' engineering time. Fabricators become involved in a project only when a designer is ready to go into production.

MOSIS is drastically reducing the cost of prototyping by holding regularly scheduled and frequent runs on which a large number of users are accommodated. Minimum lots of ten wafers are bought, but these ten wafers hold ten to thirty projects instead of only one. This means that the $30K to $50K run cost can be apportioned among those ten to thirty users.

MOSIS acquires and maintains non-proprietary design libraries fabricable by a multiple vendor base and distributes those libraries to its users and to commercial CAD vendors, who install them on their systems and make them available to users. In addition, MOSIS is developing quality-assurance procedures which, as a second neutral party, it can use to qualify wafers and provide MIL-SPEC parts. This approach allows DoD access to the whole of the U.S. semiconductor industry as opposed to just the small segment willing to undergo the rigors of conforming to MIL-SPEC certification procedures.

5.2 GOALS AND APPROACH

A few years ago, it become clear that industry would transition from the NMOS technology to CMOS. The reasons for this lay mainly with reliability of devices as their sizes shrank to the limits of available lithography. Device physics indicated that devices could be made to work with channel lengths of only 0.3 microns. A look at the expected evolution of lithography technology indicated that feature sizes as small as 0.3 microns would eventually become available. The problem that surfaced, however, was that if feature sizes became smaller and if the voltage supply levels were kept unchanged, then power density would go up. Increased power density would mean higher temperatures, which in turn would mean an increased rate at which the various failure mechanisms associated with silicon would come into play. NMOS, with its resistive loads, was recognized as a power-consuming technology that would have to be replaced with complementary MOS or CMOS technologies, which involve no resistive loads.

A second realization was that DARPA systems researchers were not going to be satisfied with accessing technology that was a few years behind the leading edge. The architectures they were interested in exploring and developing required the maximum density and performance available from technology. MOSIS' traditional approach of accessing technology that was a few years behind the leading edge would not be acceptable. Instead, MOSIS designers wanted to start exploring technologies as they were being developed. They wanted to be able to go into production as soon as development of the advanced processes was complete, instead of starting their designs at that time.

In order to prepare for the imminent switch from NMOS to CMOS and to provide the DARPA community with leading-edge technology, the MOSIS Advanced VLSI program was started.

The basic idea was to develop interfaces to groups within the semiconductor industry involved in the development of the most advanced CMOS processes. Runs would be held with selected segments of the DARPA design community submitting designs to

allow development of process-control monitors, design rules, and design styles years ahead of the common availabilty of the processes. The advanced processes that were accessed in this manner involved feature sizes of 1.2 microns.

Since its inception, this program has seen the following:

- development of interfaces to five vendors
- design of parametric test structures for 1.2-micron CMOS and development of the associated test code
- definition of design rules
- development of software for assembling 1.2-micron runs
- successful completion of eight runs

The first four runs of the Advanced VLSI program were fabricated by four different vendors. This experiment showed that only one vendor, GE, had its process sufficiently under control as to warrant its use by the MOSIS community. Now the MOSIS community is accessing that vendor's process. Since then, new vendors have claimed completion of the development of their processes and MOSIS is in the process of initiating or reinitiating experimental runs with them.

The most significant result of the program was not the successful completion of the runs but the methodology that MOSIS put into operation to allow its advanced community efficient access to leading-edge CMOS. The interaction with the various process development groups led to a sufficient understanding of the processing issues to allow MOSIS to formulate a set of design rules that are not only vendor independent but that also scale with feature size.

What scalable rules mean in a practical sense is that geometry of designs based on these rules can be scaled to run on 3-micron processes as well as 2-micron and 1.2-micron processes.

The formulation of a set of scalable rules reduced one of the most serious disadvantages associated with seeking to access the most advanced processes for the DARPA VLSI design community. Advanced processing usually carries quite a cost and turnaround penalty. The cost for prototyping is higher and the queues for processing are longer. With advanced processing, costs are driven up not so much because of the increases in cost per wafer, but mostly because the type of lithography required drastically reduces the number of users that can be put on a prototyping run. Smaller feature sizes require lithography based on "stepping" one small image containing only a small number of designs across the surface of a wafer. Larger feature sizes allow the whole wafer to be exposed at once, by an image previously composed to have a very large number of designs. Large feature-size runs can easily carry many tens of

designers, with the cost of the run shared among them. Small feature size runs can carry only a few designs. The cost of a small feature size prototyping run per design project can easily be ten times the cost per design project in a large feature size run.

Custom development for a chip usually requires several fabrication iterations. Such developments are difficult when each iteration is extremely expensive and takes a long time to complete. Some experienced systems designers insist that, because of this, it is not possible to develop complex systems based on leading-edge technology. They claim that only mature technologies should be used, because of their attendant lower costs, faster turnarounds, availability from a large number of sources, and lower risk.

Scalable rules allow advanced systems designers to do their initial chip development iterations using higher feature size technologies, where cost is low and turnaround is fast. Only in the final development phases do they use the smallest feature size technologies. There is even the possibility that by the time the the advanced technology has been accessed it will have become mature. This is the approach that members of MOSIS advanced design community are being advised to follow.

5.3 SCIENTIFIC PROGRESS

5.3.1 Scalable Design Rules

MOSIS has distributed to its community its scalable and vendor-independent design rules for 3-micron, 2-micron, and 1.2-micron p-well, n-well, and twin-tub CMOS/Bulk. These rules exist as files that can be accessed as automatic responses to messages sent to MOSIS.

The rules are in terms of a parameter lambda that assumes different values for different feature size processes. For example, lambda is equal to 1.5 microns for 3-micron processes, 1 micron for 2-micron processes, and 0.7 microns for 1.2-micron processes. The rules are drawn rules, with all geometrical relationships such as spacings, widths, and overlaps in terms of small integer values of lambda. This and the fact that all the rules fit onto one page make them easy to remember.

Two contact design layers exist: contact-to-active and contact to-poly. There are rules requiring sufficient separation between the two kinds of contacts in order to allow MOSIS to independently adjust contact overlaps and conductive layer widths when converting geometry from drawn to mask.

The ability to scale and to independently adjust active, poly, metal1, and metal2 overlap of contacts and vias, and the biasing of active, poly, metal1, and metal2

geometry allow MOSIS to fit its rules very close to that of a fabricator. Very little is lost by using MOSIS' scalable rules as opposed to those defined by a fabricator to fit its specific process.

5.3.2 Generic Process Specification

MOSIS has defined all the conventions necesary to accept from its community designs in 3-micron, 2-micron, and 1.2-micron p-well and n-well CMOS/Bulk. This includes the definition of the design layer extensions and the various ways in which the scalable CMOS/Bulk geometry can be transmitted. For example, designers can submit designs with the well and implant layers not identified with regard to type. Depending on whether the CMOS run is p-well or n-well, MOSIS then makes the well layer p-type or n-type and the select p or n. Another form of submission defines both p-well and n-well layers and p-select and n-select layers; MOSIS then selects which type of layer to ignore, depending on whether the fabrication run is to be p or n.

MOSIS has distributed to its community detailed wafer acceptance specifications to be used to accept CMOS/Bulk wafers from the majority of its vendors. MOSIS has developed process-control monitors to allow it to independently determine whether delivered wafers conform to previously agreed-upon wafer-acceptance specifications. In addition, MOSIS has distributed to its community SPICE models characterizing the transistors expected on its 3-micron, 2-micron, and 1.2-micron CMOS/Bulk runs.

5.4 IMPACT

The Advanced VLSI project was initiated in order to interface the DoD systems design community to leading-edge technologies. By working to make new VLSI technologies available to this design community, the Advanced VLSI project is significantly reducing the costs and delays involved in the design of prototype VLSI systems.

5.5 FUTURE WORK

5.5.1 CMOS/Bulk, 1.2 Micron

A number of fabricators have developed 2-micron CMOS/Bulk processes. Since 2-micron CMOS is becoming the state of the art, MOSIS is now actively pursuing expansion of its 2-micron vendor base along with expansion of its 1.2-micron vendor base. MOSIS is now establishing a regular (once a month) 1.2-micron fabrication run schedule.

5.5.2 Packaging

We will investigate more advanced packaging techniques, such as various ceramic carriers (direct inking, etc.) and plastic tape carriers. ISI is developing the necessary interface (software, frames, instructions for users, etc.) to support the use of these advanced packaging technologies by the community.

5.5.3 Other Services

As the needs of the systems design community become apparent, the Advanced VLSI project will move to meet them. Specific areas of interest in the coming year include the release of a 1.2- and 2-micron standard cell library from the same agency that developed the 3-micron library. This new library will have 'macros' for ram cells, etc., as well as SSI and MSI cells.

6. VLSI

Research Staff:
George Lewicki
Ron Ayres
Jeff Deifik
Joel Goldberg
Wes Hansford
Lee Richardson
Craig Rogers
Carl Service
Bing Sheu
Barden Smith
Jeff Sondeen
Vance Tyree

Research Assistants:
David Hollenberg
Ming Hsu
Wen-Jay Hsu
Shih-Lien Lu
Mahesh Patil
Vijay Sharma
Je-Hurn Shieh
Eric Shih

Support Staff:
Barbara Brockschmidt
Mike Curry
Sam Delatorre
Terry Dosek
Kathie Fry
Terri Lewis
Christine Tomovich

6.1 PROBLEM BEING SOLVED

The VLSI design communities of DARPA and NSF require access to VLSI fabrication in order to investigate design methodologies and architectures that use state-of-the-art technologies. Recognizing this need, DARPA established the MOSIS (MOS Implementation Service) system at ISI in January 1981. MOSIS has accomplished the following:

- reduced the cost of VLSI prototyping
- shortened turnaround time for VLSI prototyping
- freed designers from fabrication idiosyncrasies
- made design less dependent on specific fabrication lines

A cost reduction of one to two orders of magnitude has been achieved by sharing a single fabrication run among many users. Also, by centralizing (and computerizing) idiosyncratic knowledge about vendors, MOSIS minimizes the the time designers spend away from the design problem. Serving as the only interface between its design community and the vendor base, MOSIS is able to provide turnaround times of eight to ten weeks for standard technology runs, except when unusual fabrication problems occur. Nonstandard technologies and experimental runs generally require longer fabrication schedules.

6.2 GOALS AND APPROACH

MOSIS combines the various aspects of maskmaking, wafer fabrication, and package assembly. The major components of the MOSIS system are listed below.

- interaction with the designers
- handling of their design (CIF or GDS) files
- communication over either the ARPANET or GTE Telemail
- placement of projects on dies, and dies on wafers
- matching of MOSIS design rules to specific vendors' design rules, and addition of alignment marks, critical dimensions, and test devices
- fabrication of E-beam mask sets (via subcontract)
- fabrication of wafer lots (via subcontract)
- wafer probing and data analysis of MOSIS test structures
- generation of SPICE decks for each run
- generation of bonding maps
- wafer sawing, die packaging, and bonding (via subcontract)
- device distribution

Designers use any available design tools to create artwork (layout) files, which are sent to MOSIS via the ARPANET or other computer networks. MOSIS compiles a multiproject wafer and contracts with the semiconductor industry for mask making, wafer fabrication, and packaging. MOSIS then delivers packaged IC devices to the user. The user perceives MOSIS as a "black box" that accepts artwork files electronically and responds with packaged IC devices.

Though MOSIS may be instrumental in providing cells and design tools to the user, it is the sole responsibility of the user to see that the submitted patterns yield working designs. One may compare MOSIS to a publisher of conference proceedings compiled from papers submitted in "camera-ready" form, where the publisher's responsibility is to produce the exact image on the right kind of paper using the appropriate ink and binding--but not to address the spelling, grammar, syntax, ideas, or concepts of the various papers.

MOSIS provides a clean separation of responsibility for the "printing" of chips. The semiconductor manufacturer is responsible for the processing of the parts and must satisfy MOSIS's rigorous quality-control specifications. MOSIS is responsible to the user for the quality and timeliness of the fabrication. The user is responsible for the proper design of the parts and may use any design methods he finds appropriate for his needs.

It is quite common that very advanced and sophisticated chips fabricated by MOSIS work on "first-silicon." An example of this is Caltech's MOSAIC--this is an amazing accomplishment with the existing design tools. Unfortunately, this is done at a considerable cost; for example, it is estimated that Caltech's MOSAIC chip consumed over 1,000 CPU hours on various VAXes before it was submitted to MOSIS for fabrication.

6.3 SCIENTIFIC PROGRESS

6.3.1 Technology Base for Fabrication Runs

NMOS

MOSIS routinely supports NMOS at 3-micron and 4-micron feature sizes, with *buried*, rather than *butting*, contacts, in accordance with the Mead-Conway design rules.

CMOS/Bulk

MOSIS routinely supports 3-micron (feature size) CMOS/Bulk. There are two technologies available through MOSIS at 3 microns. The first is a single-metal, double-poly process, where the second poly layer serves as an electrode for high-value capacitors. The second is a double-level metal process, with runs every other week. This process has two sets of design rules. One is specific to 3-micron p-well technology; the other is a scalable set applicable to 3-micron, 2-micron, and 1.2-micron p-well and n-well CMOS/Bulk technologies. As part of the Advanced VLSI program, MOSIS now offers fabrication at 2 and 1.2 microns. MOSIS is actively engaged in expanding its 2-micron and 1.2-micron vendor base.

CMOS/SOS

MOSIS supports CMOS/SOS fabrication with 4.0 micron feature size in accordance with the Caltech design rules. All of the SOS vendors support these design rules.

Completed Fabrication Runs

The following is a categoric breakdown by technology of the fabrication runs completed during this reporting period:

21	runs	NMOS, 3/4 microns[1]
22	runs	CMOS/Bulk, 3 microns
2	runs	CMOS/SOS, 4 microns
5	runs	Wafer-Scale Integration

6.3.2 Fabrication Interface

The MOSIS vendor base has expanded substantially during this reporting period. Increased user feedback and more extensive test results have allowed the MOSIS project to determine and communicate fabrication requirements to new vendors. This has resulted in higher quality wafers and the development of consistently reliable vendor sources for mask making and NMOS fabrication.

---

[1] One run was commercially funded

MOSIS has instituted procedures to manage the vast amount of information inherent in dealing with a multi-vendor base. Many administrative tasks have been automated, including the maintenance of templates to determine fabrication requirements specific to vendor and technology (a single vendor often provides several fabrication technologies).

6.3.3 Quality Assurance/Design Interface

Most MOSIS devices are prototypes without established functional testing procedures. Generally, the designers who receive these devices are still debugging the designs, rather than checking for fabrication defects introduced by less-than-perfect yield.

MOSIS's extensive quality-assurance program is aimed primarily at the parametric level. This guarantees that the electrical properties of the wafers are within specifications established by the best a priori simulations used in the design process. Work has continued to increase the accuracy of the SPICE parameters that are made available to MOSIS users. SPICE provides simulated mathematical models for the behavior of transistors, allowing designers to assess a small digital circuit idea, to avoid faulty design, and to improve their chances of success in fabrication. The electrical criteria are a superset of the SPICE parameters at level II. They include a ring oscillator, which gives a rough idea of the speed of the resulting circuitry. The electrical properties of the wafers are extracted first by the fabricator, who uses either his own process-control monitoring devices or the MOSIS test structures. Only wafers passing these tests are delivered to MOSIS.

It is a common practice in the IC industry to save functional probing time by probing wafers in only a very few sites. This practice makes sense, because all parts are subject to functional testing and because this parametric probing serves only to eliminate disastrously bad wafers.

Since designers hand-test most MOSIS devices, MOSIS requirements for parametric testing are higher than industry standards. MOSIS inserts its own test strip on every device, if space permits. This probing provides important statistics on the electrical properties and their distribution across the wafer. Most wafers have uniform distribution: some, however, have other statistical patterns, such as significant gradients and bimodal distributions.

These in-depth statistics are available only to the fabricators. Designers receive the general statistics (mean and variance) for each run. Interested users can request the specific values of the parameters extracted near any of their chips.

Users comparing the performance of actual chips to their simulations find it useful to rerun the simulation with the actual a posteriori parameter values (SPICE deck) extracted for that run.

A perfect set of electrical parameters does not guarantee perfect yield, so there is always a need for functional testing. MOSIS does not have the facilities for high-speed functional testing, but can perform partial functional testing. This screening typically catches the majority of fabrication defects, such as shorts. The screening is performed by applying and reading user-provided vectors to each device before the wafer is cut; those failing the test will not be packaged. By screening the larger chips--which typically have lower yield and higher packaging cost, and are required in larger quantities--MOSIS significantly reduces the packaging cost.

6.3.4 Standard Pad Frame/Packaging

MOSIS's current packaging strategy is to package enough parts to ensure a 90 percent probability of delivering a defectless part to the designer. This strategy was acceptable when most of MOSIS's community was designing small circuits and the fraction of packaged defective parts was small. However, a significant portion of the community has successfully completed the development of large designs and now wants from 300 to 3,000 working parts to begin developing prototype systems based on parts obtained through MOSIS. The yield for these large designs is expected to be 25 percent at best. If MOSIS were to follow its current strategy of packaging parts without any testing to indicate functionality, it would be packaging four times the required number of parts to achieve a requested quantity.

To avoid such waste, MOSIS has worked with Stanford University to define a functional test language (SIEVE) and has developed hardware to effect the testing specified by that language. Users now have the option of submitting text describing limited test procedures to be used at wafer probe to screen out bad parts. The purpose of this screening is to detect the types of "trivial" defects that cause the majority of bad parts and, therefore, to reduce packaging costs. Full functional testing is expected to be done by the user.

For designs with custom pad layouts, it is the responsibility of the designer to provide MOSIS with the custom probe card to probe his circuits. To eliminate the inconveniences associated with generating custom probe cards for every design, MOSIS has developed a set of standard pad frames, each specifying exactly where the pads are positioned. MOSIS stocks probe cards for each of the frames.

These standard frames are also expected to facilitate packaging. Bonding diagrams for projects currently submitted are generated manually, because several attempts to automate this process have met with only limited success. Bonding diagrams instruct the packager to connect a specific pad on the chip to a specific pin on the package. Standard pad frames have standard bonding diagrams, eliminating the the need to generate a new diagram for each project. The increased use of the standard pad frames

has reduced the considerable amount of time needed to manually generate bonding diagrams. Standard frames also allow the bonding process itself to be automated. Automated, programmable bonding machines are currently available. Standard pad frames make possible a scenario in which an operator would identify a first pad and package pin; programmed information would then control the bonding on a chip.

6.3.5 Standard Cells

MOSIS has acquired from DARPA a government-designed standard cell library for 3-micron p-well CMOS/Bulk, designed in rules identical to MOSIS and thus capable of being fabricated by MOSIS CMOS fabricators.

MOSIS has made this library available to its community. Moreover, MOSIS has contacted all the major commercial CAD vendors to encourage them to support the library. A number of them have decided to do so. This allows the MOSIS community to purchase turnkey CAD systems to design standard-cell-based circuits that can be fabricated by MOSIS.

6.4 IMPACT

MOSIS's main function is to act as a single interface ("silicon broker") between a geographically distributed design community and a diverse semiconductor industry. As such an interface, MOSIS has significantly reduced the cost and time associated with prototyping custom chips.

The greatest impact of MOSIS, however, is in the community it has created. The MOSIS user community shares not only the fabrication services, but also experience, cells, tools, and software. The rapid growth of the community proves that the services provided by MOSIS are useful and important to both the academic and the R&D communities.

6.5 FUTURE WORK

With a large portion of the MOSIS community now designing in CMOS, MOSIS will continue to put considerable effort into providing low-cost and fast-turnaround CMOS fabrication.

7. KITSERV
VLSI KIT DESIGN SERVICE

Research Staff:
Robert Parker
Robert Hines
Jeff LaCoss
Richard Shiffman
Bert White

Support Staff:
Janna Tuckett
Shorty Garza
Jerry Wills

7.1 PROBLEM BEING SOLVED

The ISI-based, DARPA-sponsored MOSIS Service has demonstrated the ability to provide fabrication of full-custom integrated circuits on demand. The availability of custom fabrication has spawned considerable growth in system architecture research within the DARPA community. This interest in system design and rapid system prototyping has emphasized the need for VLSI design verification capabilities.

The *KITSERV* (a VLSI kit design service) effort will design and develop new tester system architectures to provide low-cost functional test capabilities integrated into the DARPA-sponsored, VLSI design environment. Integrated functional test capabilities will greatly enhance the ability to debug new chip designs and reduce the time involved in demonstrating "proof-of concept" development.

7.2 GOALS AND APPROACH

The overall goal of the KITSERV project is to develop cost-effective test capability for VLSI designers as an integral part of their design environments, and to make this capability available to a large population of designers by transferring the technology to commercial companies willing to market and support the testers. Specific goals involve addressing the shortcomings of commercial test systems.

Large parametric and functional testers are available in the commercial marketplace (costing approximately $1 million each). They are designed to support production testing. These systems are extremely costly to program, are not compatible with the DARPA community test interchange language, and are not integrated into the design environment. They are not cost-effective for prototype and small production testing. Lower cost (approximately $125,000) functional testers have recently been introduced. These testers run at reasonable test rates, but they have limited test capabilities, are not oriented toward concurrent testing of many different designs, and are not easily integrated into the DARPA design environment. These lower cost testers require a

hand-wired "load board" for each chip design to be tested, making them useless in the DARPA/VLSI Brokerage prototyping environment (because the MOSIS Service supports dozens of designs on each fabrication run). Currently available testers are stand-alone units with limited local test vector storage, which severely restricts their ability to sustain test rates for the complex tests required, i.e., verifying the architecture project designs being generated by the DARPA Strategic Computing researchers.

These shortcomings will be addressed by using custom VLSI chips to increase functionality while reducing tester system costs. "Load boards" will be replaced by automated pin programming and "tester per pin" archectecture. Testers will be developed to support the community test vector interchange format.

Most test systems lack the flexibility required to interface to the various design workstations used in the DARPA VLSI design community. Most systems are outgrowths of logic analyzer designs with complex and costly external pods and cables. Many test systems trade off specific performance features of interest to the DARPA CMOS designers for more general features required to capture a broader tester market. There are currently no testers available, short of the largest systems described above, that can support the inherent increase in operating frequencies that will be achieved with chips fabricated by the MOSIS Service at 1.25 micron technology. Likewise, the increased design complexity that will be achieved with wafer-scale integration, coupled with the requiremnts for hundreds of test pins and the inherent increase in operating speeds, will require very wide, very deep, and very fast local vector storage with implied data rates in the hundreds of megabytes per second, far in excess of commercially available testers.

Issues of interface flexibility will be addressed by using industry standard host system interfaces and driver software written in the C language. Increases in test system speed requirements will be addressed by implementing VLSI circuitry in smaller geometry technologies.

KITSERV will provide a planned comprehensive approach to developing tester products. As researchers will use DARPA-developed design tools on the SUN workstation, KITSERV will provide the ability to use DARPA-developed test capabilities as well.

7.3 SCIENTIFIC PROGRESS

SUNKIT ONE, the first of a family of test systems, has been completed. SUNKIT ONE involved the design and production of a VLSI chip tester based on a VLSI chip set developed for DARPA at Stanford University.

The SUNKIT ONE tester connects to a host computer, such as a SUN workstation, via a DMA board installed in the host machine backplane. Test vectors written in SIEVE (the DARPA standard interchange language) are compiled on the host machine, broken down into pin directives, and sent to the tester box for execution. The VLSI chip set in the tester interprets the pin directives and addresses the appropriate device pins. The tester is housed in a custom-designed, vacuum-formed enclosure containing device sockets for all of the MOSIS Service's standard packages. Power is supplied from a remote power supply through a connecting cord to remove all AC power from the tester box. The tester features device-power-current monitoring and limiting, and provides for testing up to 128 device pins either in a packaged part or through a cable connection to a wafer probe station. A technique of "weak-drive" is employed by the custom VLSI chip set in the tester to constantly drive all DUT (device under test) pins to logic high. DUT pins attempting to drive the tester input pins must overcome the weak-drive outputs of the tester. Test speeds with SUNKIT ONE are approximately 10 microseconds per device pin directive.

The SIEVE compiler has been converted to the C language and is now available for UNIX machines, as well as the original VAX VMS implementation. The test environment is fully integrated into the design environment. A designer running the Berkeley design tools on a SUN workstation can design, simulate, submit designs to the MOSIS Service, and functionally test chips, all in the same integrated environment.

Twelve SUNKIT ONE testers were completed and distributed to the DoD community as no-cost prototype loaners for evaluation. SUNKIT ONE testers have been used to debug the MIPS-X processor developed at Stanford University and to provide wafer-level functional screening of the VLSI chips used in the Butterfly development effort at Bolt Beranek and Newman, Inc. A second effort at Stanford combined the SUNKIT ONE with a behavioral simulator to allow real-time matching of the test results to the simulator outputs. SUNKIT ONE testers are operational on VAX computers, on Multibus SUN, VME-Bus SUN, IBM-PC/AT, and Microvax workstations, and on the PDP 11/23 computer. No commercially available tester provides this level of flexibility in host support.

7.4 IMPACT

The level of tool integration achieved by merging the SUNKIT ONE into the Berkeley design environment has provided the DARPA community with a totally new system approach to VLSI design and verification. This effort demonstrated the concept of a low-cost environment that would greatly enhance the productivity of VLSI designers by reducing prototype development time. The SUNKIT ONE product, currently in use at several DARPA research sites, demonstrated the concept of system integration but fell short of providing vector test rates to match chip operating speeds attainable with the evolving MOSIS Service fabrication technologies.

7.5 FUTURE WORK

The design of SUNKIT TWO will address several shortcomings identified during the prototype evaluation of SUNKIT ONE. The most serious shortcoming was the ineffective use of host memory bandwidth. DUT pin directives were transferred individually from the host to the VLSI-based controller in SUNKIT ONE, which interpreted and executed instructions "on the fly." Since many pin directives were required per test vector, the resulting vector test rate was prohibitively slow. In addition to the data efficiency problem, the problem of data throughput was also identified. Host workstations can support a maximum continuous data rate of several megabytes per second through a DMA interface from the largest internal memory buffer of several hundred kilobytes. Data rates required to support the testing of a 64-pin DUT at 10 megavectors per second, assuming 3 data bits per pin per vector, is 240 megabytes per second. This simple calculation gives strong support to the requirement for a local vector memory in the tester. Such memory will allow the high vector rate testing without taxing the host memory bandwidth. Evaluation of these shortcomings and results of continued discussions with researchers in the DARPA community will be formative in the development of specifications for SUNKIT TWO.

SUNKIT TWO will be microprocessor based, to allow increased flexibility in host interfacing by providing a serial interface and a generic parallel interface. It will provide upward compatibility with existing SUNKIT ONE users through the same DMA interface. In addition to the host interface, the local microprocessor will control parameter setup, data handling, post-processing, and error detection. A large local vector and error buffer will be used to facilitate testing complex devices at high vector rates. The local vector buffer will use a "don't care" mask per vector, per DUT pin to support total flexibility in error reporting. A local DMA controller will be included to handle the data flow between the vector buffer and the pin-drive electronics. An optional relay board is planned to allow power and ground to be applied to any DUT pin under program control, thus avoiding the requirement for mechanical patching. The tester will be designed in a modular fashion, with pin-drive boards and relay boards interconnected in a stack to allow expansion capability for testing more DUT pins. The tester boards will be designed around a hollow center core where the DUT adapter is supported, thus providing short and controlled interconnection paths from the DUT to the drive electronics. Vector test rates are to be controlled by a programmable synthesized clock that will generate frequencies from 20 kilohertz to 10 megahertz in 10 kilohertz steps. The tester will be self-contained in a custom enclosure with its own power supply. The 80286 microprocessor will be used in the tester design as the test set-up and control mechanism and will be packaged in the form of an IBM-PC/AT-compatible mother board. Custom boards required for the SUNKIT TWO will be plugged into a backplane attached directly to the IBM-PC/AT mother board. This packaging technique will allow several options to be exercised in the use of SUNKIT TWO. First, the tester can be used in stand-alone mode with a dedicated internal IBM-

PC/AT mother board. Second, the tester can be used as an accessory to an existing IBM-PC/AT by attaching cables from the IBM-PC/AT to the SUNKIT TWO backplane, thus eliminating the internal mother board. Third, the tester can be used in unpackaged form, perhaps with just the pin-drive boards mounted directly to a probe station and attached through cables to the remote IBM-PC/AT mother board. IBM-PC/AT compatibility will allow use of the DOS operating system and the rich environment of software and hardware development tools available. The IBM-PC/AT will provide a very powerful, low-cost development platform for SUNKIT TWO.

The DUT pin drive in SUNKIT TWO will be provided by custom CMOS VLSI chips capable of driving 16 DUT pins through tri-state I/O pads. A 32-bit data path to the vector memory is planned with 2-bit (read/write, polarity) control for each DUT pin. Pin-drive chips will be phase-locked to the master clock rate generator to minimize signal skews across DUT pins and to allow all critical clock signals to be generated and used on-chip. The pin-drive chips will have an internal phase-lock loop (PLL) and programmable tapped delay line to allow strobe edge skewing under program control. Two independent pulse generators are planned on-chip to be used as DUT clocks that can be programmed to any DUT pin. These clocks are to be generated by the PLL circuitry with programmable phase, delay, and duty cycle. The clocks will not only be used as DUT clocks, but will also be modulated by the data stream to produce NRZ, RZ, and R0 data modulation types. An independent read data strobe will be generated to latch the data from the DUT, phase-skewed under program control.

We plan to design and develop several SUNKIT TWO prototypes and to distribute them for evaluation. Software development will focus initially on the basic driver and SIEVE compiler, with enhancements for tester setup parameters to support the release of evaluation prototypes. Both hardware and software development will continue during the evaluation period, to develop hardware options and to enrich the user-level software.

Longer range plans include the development and support of an enhanced vector interchange format for the DARPA community and continued enhancements to the tester performance to parallel the development of new fabrication technologies and system development activities. Industrial partners will be actively sought as commercially viable product prototypes are developed.

8. EMPIRICALLY VALID KNOWLEDGE REPRESENTATION

Research Staff:
Robert Mac Gregor
Raymond Bates
Gabriel Robins

Support Staff:
Heidi Julian

8.1 PROBLEM BEING SOLVED

A dominant theme of much current AI research is the importance of explicit, well-founded models of knowledge. Despite the fact that we now have a fairly sound understanding of the issues involved with network-based knowledge representation methodology, there is no appropriate knowledge representation tool widely available to the AI research community. Currently, each research project with knowledge representation needs must expend excessive energy building and maintaining its own, usually ad hoc and unprincipled, knowledge representation system. To further complicate the problem, availability of a knowledge base mechanism is only part of the problem: tools to use the underlying formalism must also be available. Our own experience, which involves many man-years of modeling, has pointed out the need for a usable knowledge representation system.

A usable knowledge representation system can have as significant an impact on the knowledge engineering community as formalized databases had on the information processing community. When ad hoc procedures are converted to formalized ones, users are more productive due to increased automation and error elimination. Formalized systems also facilitate the sharing of information. A usable knowledge representation system can provide a first-level standard for sharing knowledge across domains.

The usability of a knowledge representation system is derived from four attributes.

1. **It must be expressive.** The modeling tool must be able to capture the semantics of the domain in a concise, well-defined, and easy-to-use notation.
2. **It must be principled.** The modeling system must have well-defined semantics so that general-purpose reasoning agents can be supplied to manage the knowledge base as it grows and to provide useful domain-independent inferences.
3. **It must be cooperative.** The modeling system must have an appropriate set of definition and analysis tools that cooperate with the modeler to lessen the task.
4. **It must be available.** There must be a robust, efficient, and well-documented implementation of the knowledge base and modeling environment produced for appropriate hardware and software.

While researchers have *experimented* with various methodologies and produced *experimental* languages and environments, no one has yet produced a usable knowledge representation system that scores well with respect to the usability criteria mentioned above. The goal of this project is to extend the capabilities of an existing knowledge representation system, NIKL [3], to fill this need.

8.2 GOALS AND APPROACH

Our work in knowledge representation is motivated by a desire to build a principled knowledge representation system, NIKL, that can be used to provide terminological competence in a variety of applications. To this end, we have solicited use of the NIKL system in the following applications: natural language processing, expert systems, and knowledge-based software. Our research methodology is to allow application needs, rather than theoretical interests, to drive the continued development of the language. This methodology has allowed us to perform an empirical evaluation of the strengths and weaknesses of NIKL. It has also helped us identify general requirements for terminological reasoning in intelligent systems.

We classify the improvements that we have made or plan to make into three broad categories:

1. **Expressiveness:** enhancements to the *terminological competence* represented in NIKL and the inferences NIKL can make regarding the subsumption relationship.
2. **Environment:** enhancements to the tools that accompany NIKL for both maintaining knowledge bases (knowledge acquisition) and reasoning about the terminology defined in the knowledge base.
3. **Support:** enhancements to user documentation and to the reliability and availability of the implementation.

8.3 SCIENTIFIC PROGRESS

We have made several enhancements to NIKL that were in direct response to applications needs. First, we have upgraded the status of roles (binary relations) within the NIKL framework to that of first-class objects, by placing them in a separate relation hierarchy. The presence of the relation hierarchy permits us to design a more complete inference procedure with respect to roles.[1]

We have developed an integrated set of acquisition tools in a window-based workstation environment. The tools include a graph of the concept hierarchy, an EMACS editing window, and a LISP interaction window. Within the LISP interaction

---

[1] These features are discussed in detail in [3].

window, the environment can produce highly formatted ("pretty-printed") descriptions of concepts. The atoms in these formatted displays (which refer to concepts and relations), as well as the nodes of the graph and the text in the edit buffer, are all mouse sensitive and known to be NIKL constructs by the environment. This allows the user to move from one window to another in a coordinated way. It also allows the user to refer to a NIKL object simply by pointing at it in any of the various views of the network.

We have ported NIKL from Interlisp to Common LISP, and have experimented with its use on a variety of workstations and mainframe implementations of Common LISP. The integrated acquisition environment depends on some specific tools found in the Symbolics ZETALISP environment. We are actively pursuing the development of similar facilities that rely on a Common LISP implementation of a form and graphics package that requires only modest customizations for various graphic environments. A portable grapher subsystem has emerged as a by-product of this effort [7].

Finally, we have produced a user manual for NIKL [6], which serves both to make NIKL more accessible to users outside of ISI and to set a standard for the community demonstrating what the capabilities of a knowledge representation system ought to be.

8.4 IMPACT

The development of NIKL has led to a rapid increase in the number of research efforts using KL-ONE-like knowledge representation languages. The Consul user interface management effort was the first to employ the language. The basic Consul user interface used models to structure displays [4]. NIKL was used here to supply semantics to terms used in a forward-chaining inference system. Consul also used NIKL in a natural language case-frame parser, where the model represented some of the lexical semantics as well as the case frames [8]. User interface research has continued with the use of NIKL in a multi-modal user interface system, II [1]. Natural language research has seen NIKL applied in a text-generation system [9] and another natural language understanding system [11]. In the two natural language efforts, the systems were able to share a common world model through a NIKL knowledge base. Parallel applications have occurred in the area of expert systems. The Explainable Expert Systems effort used NIKL to describe the problem domain in an effort that produced an expert system creation system [5].

One of the healthiest signs of NIKL's success is the number of knowledge representation researchers who have used NIKL as a starting point for their work. For example, KL-TWO uses NIKL as a T-Box to which it added an A-Box [10], and Ron Brachman has related that NIKL is the standard by which he has measured the Krypton T-Box [2].

8.5 FUTURE WORK

We are planning to extend NIKL along three different dimensions. The first of these might be called *flexibility*. Currently, a loaded NIKL model can be extended, but not modified; i.e., to *change* an individual concept specification in an already loaded knowledge network, one must edit the specification and then reload the entire model. We are developing an incremental classifier that will permit editing of already loaded knowledge networks. This extension will make the process of designing and debugging NIKL models much easier.

Second, we plan to significantly extend the representational capabilities of the system, providing specialized syntax and inference methods for sets, sequences, arithmetic intervals, inverse and transitive relations, and necessary and sufficient conditions.[2]

Finally, we plan to add to NIKL a pattern-matcher and a truth-maintenance subsystem. Together, these new inference mechanisms accomplish the integration of terminological knowledge, represented as a NIKL model, with assertional knowledge stored in a database.

The completed system will exhibit a much broader range of capabilities than are currently available. Not only will the extended NIKL increase the knowledge representation support available to current NIKL users, but it will open up NIKL to use by new and different applications.

REFERENCES

1. Arens,Y., L. Miller, and N. K. Sondheimer, Presentation Planning Using an Integrated Knowledge Base, 1987. Submitted for publication.

2. Brachman, R. J., V. P. Gilbert, and H. J. Levesque, "An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 532-539, Los Angeles, August 1985.

3. Kaczmarek, T., R. Bates, and G. Robins, "Recent developments in NIKL," in *AAAI-86. Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-86-167, October 1986.

4. Mark, W., "Representation and inference in the Consul system," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, IJCAI, 1981.

---

[2]Many of these features are described in [3].

5. Neches, R., W. R. Swartout, and J. Moore, "Explainable (and maintainable) expert systems," *IEEE Transactions on Software Engineering* SE-11, (11), 1985, 1337-1351.

6. Robins, G., The NIKL Manual, USC/Information Sciences Institute, April 1986.

7. Robins, G., "The ISI Grapher: A portable tool for displaying graphs pictorially," in *Symbolikka '87*, Helsinki, Finland, August 1987. Also published as USC/Information Sciences Institute, RS-87-196, September 1987.

8. Sondheimer, N. K., R. M. Weischedel, and R. J. Bobrow, "Semantic interpretation using KL-ONE," in *Proceedings of COLING84*, pp. 101-107, Association for Computational Linguistics, July 1984.

9. Sondheimer, N. K., and B. Nebel, "A logical-form and knowledge-base design for natural language generation," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-87-169, November 1986.

10. Vilain, M., "The restricted language architecture of a hybrid representation system," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, August 1985.

11. Weischedel, R., D. Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked, and D. Stallard, *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program*, Bolt Beranek and Newman, Inc., Technical Report 6522, June 1987.

9. TEXT GENERATION FOR STRATEGIC COMPUTING

Research Staff:
William Mann
Norman Sondheimer
Mark Feber
Christian Matthiessen

Research Assistants:
Susanna Cumming
Shari Naberschnig

Support Staff:
Lisa Trentham

9.1 PROBLEM BEING SOLVED

The U.S. military is an information-rich, computer-intensive organization. It needs easy, understandable access to a wide variety of information. Currently, information is often expressed in obscure computer notations that cannot be understood without extensive training and practice. Although easy discourse between the user and the machine is an important objective for any situation, this issue is especially critical in regard to automated decision aids such as expert-system-based battle management systems that must carry on a dialog with a force commander. A commander cannot afford to miss important information, and it is unreasonable to expect force commanders to undergo highly specialized training in order to allow them to understand obscure computer dialects that change from machine to machine.

A great deal of work has been done in the area of natural language understanding, and this work is starting to pay off with the delivery of functional interfaces that interact naturally with the user. Comparatively little, however, has been done in the area of natural language *generation*. There is currently no effective technology for expressing complex computer notations in ordinary English. If there were, computer-based information could be made more accessible and understandable in a way that was less subject to personnel changes.

9.2 GOALS AND APPROACH

This project is creating and demonstrating new technology to provide an English-in, English-out interface to computer data, called Janus. ISI is developing the English-out (text generation) portion of Janus, referred to as Penman. Initial capability will be demonstrated on a naval database, but most of the techniques are more general and will be able to be reapplied to other problems. The end result will be a system that produces answers to queries and commands in the form of text that is understandable to any user who understands English.

Both understanding and generation components are necessary in an effective

information system interface. It is also necessary that the technologies used by these two components be absolutely consistent. If they are not, an embarrassing situation can occur in which the interface fails to understand a piece of text that it has just output. This type of situation could cause users to lose confidence in the system.

In order to alleviate this problem, we are developing Penman in close cooperation with a natural language understanding project at Bolt Beranek and Newman, Inc. (BBN). This cooperation extends to joint development of the knowledge representation software and the lexicon design, as well as work to insure compatibility between BBN's RUS understanding grammar and ISI's Nigel generation grammar.

The understanding/generation compatibility effort can be considered the first of our goals for the Penman system. Another is the ability to generate substantial amounts of English text, up to the multiparagraph level. A useful interface should not be restricted to short replies or comments.

A third goal is to make Penman as portable and domain independent as possible. There are many knowledge domains available for interface systems, and few specialists to create the interfaces. Thus, we are emphasizing domain-independent capabilities over domain-dependent ones.

Finally, there are many linguistically and computationally significant issues that must be addressed if high-quality generation is to be achieved. For example, we are working on improved methods for knowledge representation, for vocabulary development, and for using semantics and discourse context to guide the generation of text.

9.3 SCIENTIFIC PROGRESS

This project was put in place at the beginning of FY85 (October 1985) in order to develop the first natural language generation capability robust and capable enough to be used in DARPA's Strategic Computing program. It is intended that this interface be coupled to the battle management system being developed under DARPA's Fleet Command Center Battle Management Program (FCCBMP). In the first year of the effort, ISI has designed the first generation system to produce English from output demands in formal mathematical logic using a broad-coverage grammar and an artificial intelligence knowledge base, and demonstrated its use with the generation of a set of sentences. Because there is no existing knowledge database for the Naval domain, we applied Penman to an existing knowledge base that was developed by the Consul project, a previous DARPA-sponsored project at ISI [5]. The knowledge domain of the Consul database is electronic mail and calendars.

As part of the Penman design, this project has created and delivered a Master Lexicon

facility, ML, for vocabulary acquisition and use [1, 2, 3]. ML is unusual in that it is compatible with the two radically different grammars of English in Janus: the Nigel generation grammar and the RUS understanding grammar. The system features a multi-window, bitmapped display that can be manipulated with keyboard and mouse. Online documentation is provided for all lexical choices. ML is built to allow for the extension of the Janus lexical system. It is also possible to use ML with other natural language processing systems by replacing a single data structure.

In addition, work has been done on bringing the Nigel and RUS grammars into compatibility.

We have as a general goal the development of natural language generation capabilities. Our vehicle for these capabilities is a reusable module designed to meet all of a system's needs for generated sentences. The generation module must have an input notation in which demands for expression are represented. The notation should be of general applicability. For example, a good notation ought to be useful in a reasoning system. The notation should have a well-defined semantics. In addition, the generator has to have some way of interpreting the demands. This interpretation has to be efficient.

In our research, we have chosen to use formal logic as a demand language. Network knowledge bases will be used to define the domain of discourse in order to help the generator interpret the logical forms. A restricted, hybrid knowledge representation will be used to analyze demands for expression using the knowledge base. We have developed a design that calls for the following:

1. developing a demand language, Penman Logical Form (PLF), based on first-order logic [7];
2. structuring a NIKL (New Implementation of KL-ONE) network to reflect conceptual distinctions observed by functional systemic linguists;
3. developing a method for translating demands for expression into a propositional logic database;
4. employing KL-TWO [8] to analyze the translated demands;
5. using the results of the analyses to provide directions to the Nigel English sentence-generation system [6].

PLF is essentially complete. To first-order logic, PLF adds restricted quantification, i.e., the ability to restrict the set quantified over. In addition, we allow for equality and some related quantifiers and operators, such as the quantifier for "there exists exactly one ..." ($\exists!$), and the operator for "the one thing that ..." (ι). We permit the formation and manipulation of sets, including a predicate for set membership (ELEMENT-OF). We also have some quantifiers and operators based on Habel's η operator [4].

The system's knowledge has been organized in a new way in order to facilitate English expression. In order to support both the inferences necessary for parsing and question-answering and the classificatory reasoning necessary for generation, we are developing a very general, domain-independent taxonomy of the types of entitites, relations, and actions that occur in the world and that determine the distinctions made in English grammar, and then subordinating the particular domain model to that. This taxonomy, which we call the Upper Model, is implemented in the NIKL knowledge representation language, whose property inheritance facility enables the subordination to work as desired. Because abstract concepts corresponding to the major conceptual categories of English (such as process, event, quality, relationship, and object) are defined in this way, fluent English expression is facilitated.

NIKL contains only definitional information. In order to represent instantial information as well, we link another reasoner, PENNI, to NIKL, and call the whole system KL-TWO. KL-TWO is a hybrid knowledge representation system that uses NIKL's formal semantics and PENNI's propositional logic reasoning ability. We translate a logical form into an equivalent KL-TWO structure. All predications appearing in the logical form are put into the PENNI database as assertions. A separate tree, reflecting the variable scoping, is created. Separate scopes are kept for the range restriction of a quantification and its predication.

The Nigel grammar and generator embody the functional systemic framework at the level of sentence generation. Within this framework, language is viewed as offering a set of grammatical choices to its speakers. Speakers make their choices based on the information they wish to convey and the discourse context in which they find themselves. Nigel captures the first of these notions by organizing sets of minimal choices into systems. The grammar is actually just a collection of these systems. The factors the speaker considers in evaluating his communicative goal are shown by questions called inquiries inside of the chooser that is associated with each system. A choice alternative in a system is chosen according to the responses to one or more of these inquiries. It is these inquiries that we have implemented.

Our implementation of Nigel's inquiries, which uses the connection and scope structures with the NIKL upper structure, is fairly straightforward to describe. Since the logical forms reflecting the world view are in the highest level of the NIKL model, the information decomposition inquiries use these structures to do search and retrieval. With all of the predicates in the domain specializing concepts in the functional systemic level of the NIKL model, information characterization inquiries that consider aspects of the connection structure test for the truth of appropriate PENNI propositions. The inquiries that relate to information presented in the quantification structure of the logical form search the scope structure. Finally, to supply lexical entries, we associate lexical entries with NIKL concepts as attached data and use the retrieval methods of PENNI and NIKL to retrieve the appropriate terms.

Besides the technology described above, work has been done on aspects of discourse modeling. This included a study of object description and an initial noun phrase planner.

In order to produce the sorts of larger texts that are needed in a User Assistance facility, work is under way to apply Rhetorical Structure Theory to text planning. We have developed a procedure that can design structures for paragraph-length texts given information about the text goal and communication situation.

9.4 IMPACT

Among the research objectives of this project are to provide a useful and theoretically motivated computational resource for other research and development groups and for the computational community at large, and to provide a text-generation system that can be used routinely by system developers. It is our intention to make Penman available to other research projects as it is further developed. This "reusability" is possible because of Penman's domain independence: Penman offers a foundation on which to build many types of natural language interfaces.

Toward the end of sharing the Penman system, we believe that publishing the results of our research and demonstrating the system as it develops are of primary importance. In January, Penman came to first fruition, generating a collection of sentences in the mail and calendar domain. A paper describing the modeling, logical representations, and generation process will be presented at the AAAI conference in August. Penman was demonstrated at the confence of the Association for Computational Linguistics, the primary conference of our community, at Columbia University in New York City in June. In addition, we have begun to write documentation for the Nigel grammar, with the intent of eventual distribution of the system. As Penman develops as a system, this type of demonstration will become more common, and we intend to collaborate with other members of the community for the purpose of using Penman and Nigel in a variety of domains.

9.5 FUTURE WORK

In the next phase of the work, we will turn our design into a working system.

Table 9-1 shows a series of independent sentences, illustrating the level of performance for which Penman is being developed. In the absence of a naval knowledge base, we are testing our ideas on the Consul knowledge domain of computer mail and calendars [5]. We have arranged and labeled the sentences to show how they might enter into computer database dialogue. Paraphrases in <brackets> show corresponding sentences of the sort that will be generated from Naval domain knowledge.

Table 9-1: Sentences from Penman

1. User: What is the read status of the message which was sent by Jones? <What is the readiness of the ship deployed by CINCPACFLT?>
2. System: Message 40 is unread. <The Nimitz is C4.>
3. User: Are any messages which were sent by Jones in the tape file? <Are any ships deployed by CINCPACFLT in the Atlantic fleet?>
4. User: Are any messages which were sent by Jones in the old ISI tape file which Smith has? <Are any ships deployed by CINCPACFLT in the new battle group which Spivak has?>
5. System: Yes. <Yes.>
6. System: Do you want me to display the read status of the messages? <Do you want me to display the readiness of the ships?>
7. User: No. <No.>
8. System: There are messages which were sent by Jones. <There are ships which were deployed by CINCPACFLT.>
9. System: However, the tape is in use. <However, the battle group is en route.>
10. User: Show me Message 40. <Show me the JFK.>
11. User: Did Smith send any messages? <Did the JFK send any CASREPs?>
12. System: Message 45 was sent by Smith. <CASREP 45 was sent by the JFK.>
13. User: Did Smith schedule his meeting with me in New York? <Did the JFK schedule its port visit in San Diego?>
14. System: Yes. <Yes.>
15. User: What does the window show? <What does the map show?>
16. System: The window shows a message which was sent by Jones. <The map shows a sail plan which was sent by CINCPAC.>
17. System: The window shows a request by Jones. <The map shows a request which was sent by CINCPACFLT.>
18. User: Did I create the window? <Did I create the map?>
19. System: The window was created when you asked me to display Message 50. <The map was created when you asked me to display Sail Plan 33.>
20. System: You deleted the message, then you asked me to display it. <You deleted Sail Plan 33, then you asked me to display it.>

Penman will be delivered to BBN, integrated into Janus, and demonstrated in an extensive series of laboratory exercises. We hope to demonstrate the combined Janus system early in 1987. This version of Penman will support coordinated English input and output, paraphrasing, and user assistance. In addition, we intend to begin the

implementation of a system to plan multisentential paragraphs in the near future. This is a logical next step in the evolution of Penman, from sengle-sentence generation to providing full-fledged text generation.

REFERENCES

1. Cumming, S., *Design of a Master Lexicon*, USC/Information Sciences Institute, RR-85-163, February 1986.

2. Cumming, S., and R. Albano, *A Guide to Lexical Acquisition in the JANUS System*, USC/Information Sciences Institute, RR-85-162, February 1986.

3. Cumming, S., *The Lexicon in Text Generation*, USC/Information Sciences Institute, RR-86-168, 1986. Presented at The Workshop on Automating the Lexicon, Pisa, Italy, May 1986.

4. Habel, C., "Referential nets with attributes," in Horecky (ed.), *Proceedings of COLING-82*, North-Holland, Amsterdam, 1982.

5. Kaczmarek, T., W. Mark, and N. K. Sondheimer, "The Consul/CUE interface: An integrated interactive environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983. Also published as USC/Information Sciences Institute, RS-83-126, April 1984.

6. Mann, W. C., and C. M. I. M. Matthiessen, *Nigel: A systemic grammar for text generation*, USC/Information Sciences Institute, RR-83-105, February 1983.

7. *Penman's Logical Language*, USC/Information Sciences Institute, Technical Manual, 1985.

8. Vilain, M., "The restricted language architecture of a hybrid representation system," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, August 1985.

10. COMMERCIAL MAIL

Research Staff:
Dale Chase
Craig Ward

Support Staff:
Manon Levenberg
Glen Gauthier

10.1 PROBLEM BEING SOLVED

The evolution of large electronic mail systems testifies to the increasing importance of electronic mail as a means of communication and coordination throughout the scientific research community. These systems include the DARPA Internet mail system, the GTE Telemail system, the MCI-Mail system, and the IEEE Compmail system. Until now these systems have operated autonomously, and no convenient mechanism has existed to allow users of one system to send electronic mail to users on another system. The Intermail system, developed by the Internet Concepts Research project at ISI, demonstrated a mail-forwarding system that allows users to send electronic mail across such mail system boundaries. The Commercial Mail project will convert this system from a research project into a commercial product.

10.2 GOALS AND APPROACH

The most significant limitation of the Intermail system is its inability to handle forwarding to more than one "other" mail system in a single interaction--that is, each message can be delivered to only one other mail system. We will eliminate this limitation by introducing a new syntax for addresses. Any number of addressees may be specified on any number of "other" mail systems. This will allow users on "other" mail systems to be included in mailing lists along with DARPA Internet recipients, UUCP recipients, CSNET recipients, etc.

It is presently possible for DARPA Internet users to communicate easily with users of both UUCP and CSNET via cooperating hosts that maintain connections with these communities. Addresses are specified in the local syntax of each of the three systems and conversions are handled by the forwarding hosts. This is the way we intend to have the commercial mail forwarding facility function. This is a relatively easy facility to provide on the DARPA side, where the address syntax is well understood, flexible, and the entire process will be under our control. On the commercial side, we will continue to use the techniques developed by the Internet Concepts project, which use forwarding information embedded in the text portion of the message. Our conversion system will, however, make it possible for DARPA Mail recipients to reply directly using existing DARPA Mail composition programs.

10.3 SCIENTIFIC PROGRESS

Upon the successful development of the Intermail system, the ISI computer center software group was tasked to build a robust, production-quality, operating Commercial Mail Relay (CMR) system. This system currently bridges the GTE Telemail system to the Internet. The current CMR system software runs under 4.3 bsd UNIX with full NFS support. Production versions of the software are running on a VAXStation II, a Microvax, and a VAX-11/750. The current system that interfaces to the Telemail system simulates a human by using a terminal to enter the Telemail system. Negotiations are under way with GTE to update their software to allow direct computer access. Because the GTE system assumes that a human is entering the system, mail errors (such as misaddressed mail) are handled in a very verbose format. This makes error handling in CMR very challenging. We currently handle some failed mail situations, such as incorrect commercial mail system, incorrect commercial mail user name, and incorrect commercial mail host, and we plan to include others.

In its first year the Commercial Mail project spent a great amount of time evaluating the structure and internals of the Intermail research project software, and building the foundation of a commercial product. During this development time, numerous demonstrations of the system were given, enabling researchers at ISI to give the Commercial Mail development team positive feedback on the development of the system. The Commercial Mail development team also developed a systems specification document. Work continues on refining and expanding the Commercial Mail system.

10.4 IMPACT

The availability of high-performance personal workstations and dedicated special-purpose processors, and the preference for these machines by researchers, have made the acquisition and operation of large mainframes capable of providing reliable mail service unacceptably high. Since most of these research machines cannot provide mail service, the Commercial Mail implementation will allow the users of these machines to remain accessible to the rest of the research community without having to individually maintain two distinct computing facilities.

The availability of a reliable high-capacity bridge between commercial mail systems and the DARPA Internet will provide better communication among contractors, especially those involved in the Strategic Computing Program who might otherwise be denied convenient access to colleagues.

10.5 FUTURE WORK

ISI will continue to update and support the CMR system. Over the three-year contract period, we will add software to the system that will allow mail to be relayed from the Internet to ONRMail, IEEE Compmail, NSFMail, MCI-Mail, Compuserve, GENIE, and AT&T Mail. ISI will also provide the following enhanced services:

- negotiate better billing procedures from commercial networks
- negotiate the implementation of new machine interfaces with commercial mail networks
- improve systems reliability
- implement software that will allow faster file transfer
- improve systems accounting
- maintain a list of known bridges to other networks such as BITNET, CSNET, UUCP, and Usenet.

The improved systems accounting software will include the facility to recharge costs to remote sites that use the CMR system.

ISI will develop software for IBM-PC-type computers and Apple MacIntosh II-type computers as part of this contract. The PC Mail software will be designed to allow access to a commercial mail system from the PC directly via modem or through a UNIX-based CMR system. The interface to the UNIX-based CMR system will be via Ethernet TCP/IP connection. The PC Mail software for IBM-PCs and Mac IIs will be similar to MH-based mail programs that run on the SUN workstations (for example, VMH and HM). Systems that interface to a UNIX-based CMR system will use the Post Office Protocol for mail transfer. One strong point of this design will be that the PC Mail software will be based on the Epsilon editor, which is an implementation of Emacs for the IBM-PC. Emacs is widely used within the DARPA community and will enable users not familiar with MH-type mail systems to access a very powerful mail system with minimal training.

Commercial mail systems have been designed with the idea that the typical user will interface to it with an IBM-PC-type computer that has terminal emulation software and file transfer capability. This type of system allows users to access their mail and send mail to other users while being charged for connect time. We believe that the majority, if not all, of the file editing and mail message review process can be handled locally on the PC and that the user need only connect to the commercial mail network for sending and receiving mail messages, turning the commercial mail network into a pseudo mail server. This will directly lower user costs, because users will be charged for significantly less connect time by the commercial networks; it will also give them access to more powerful editing and mail management software. Commercial networks will not evolve to this type of architecture, because they make their money on connect time. Another

advantage to having the majority of mail processing done locally is that messages can be queued up during the day and sent to the mail network in non-prime-time hours, lowering systems operating costs even more.

Once this software is developed, ISI will serve as a distribution center for outside agencies that would like to use this type of mail relay software. This software will be supported by a full set of user documentation. ISI will maintain an electronic mailbox that can be used by remote users to report bugs, problems, etc. ISI will also distribute the CMR software to other sites that run 4.3 bsd UNIX.

11. COMPUTER RESEARCH SUPPORT

Director: Ronald Ohlander

Technical Staff:

Software:
Dale Chase
David Bilkis
Dwight Fromm
Bryan Howard
Jim Koda
William Moore
Koji Okazaki
Rod Van Meter
Craig Ward
Tom Wisniewski

Hardware:
Daniel Pederson
Val Fucich
Glen Gauthier
Ramon Gonzalez
Fred Grolle
Raymond Mason
Daniel Trentham

Operations and Network Services:
Stephen Dougherty
Walt Edmison
James Hurd
Roylene Jones
Joe Kemp
Roger Lewis
Sean Schur
Toby Stanley
Christine Tomovich
Mike Zonfrillo
Vicki Gordon
Kevin Haley
Anna-Lena Neches
Chris Refuerzo
Wayne Tanner
Robert Wormuth

Support Staff:
Manon Levenberg
Nancy Garman
Pat Thompson

11.1 PROBLEMS BEING SOLVED

The Computer Research Support project is responsible for providing reliable computing facilities on a 24-hour, 7-day schedule to the ARPA-Internet research and development community. At the same time, the project makes available to ARPA-Internet users the most current upgrades and revisions of hardware and software on the supported machines. The project provides continuous computer center supervision and operation, and a full-time customer-service staff that is responsive to user inquiries. This project primarily supports a major computer installation at ISI's main facility in Marina del Rey, but shared staff within the facility lends infrastructure support to ISI's internal research efforts.

11.2 BACKGROUND

ISI provides cost-effective and key computer support to researchers at ISI, DARPA contractors and affiliates, and the military services. In addition to providing raw computing service on TOPS-20 to members of the DARPA research community and DARPA itself, ISI also originally developed and provided and/or now helps to support many additional mature software packages for the entire ARPA-Internet community.

including text editors (XED, Emacs), mail handlers (Hermes, MSG, SNDMSG, MM), compilers (PASCAL, C, FORTRAN, COBOL, MACRO, BLISS), language environments (Interlisp, Mainsail, Ada), and network access tools (Telnet, FTP, SMTP).

ISI has built a reputation for excellence and efficiency in both experimental computer services and production services, and has repeatedly demonstrated its continuing high standards to users while maintaining a relatively small and expert staff. ISI provides guidance to DARPA on what constitutes a sensible load for a TOPS-20 host, and also provides management control to ensure an adequate level of support for the DARPA user community.

ISI has also shown itself to be extremely proficient in the realms of network, access, and host security, successfully walking the fine line between protecting the interests of the community and having that security be an encumbrance to the user community. Security involves a number of access controls and, perhaps more important, it includes protection schemas, password encryption checkers, password breakers/testers, and monitoring demons that can be activated when a problem is suspected.

ISI and its current research projects have benefited substantially from in-house competence in computer services, particularly as the original support of the TOPS-20/DEC-KL environment has widened to include an eclectic and complex collection of hardware and software architectures. The staff of the Computer Center has grown in expertise and maturity to meet the demands of this wider collection of equipment and software, while the total head count of the group has been reduced.

With the continuing evolution of computer workstation technology, ISI expects to see a decline in the demand for network-based, interactive, timesharing cycles as users move into workstation/local area network environments.

11.3 GOALS AND APPROACHES

The ISI Information Processing Center provides support in four distinct, though tightly interrelated, areas: Hardware, Systems Software, Operations, and Network Services. The overall charter of the organization is to assure that the needs of the user community are addressed and protected as efficaciously as possible. To achieve this end, each group is concerned about the effective use of the machines and software tools, and about the security of the physical plant, the system files, and other online information. The more specific goals and approaches of each group are summarized below.

Hardware

To achieve a reliability goal of 98.7 percent scheduled uptime, preventive and remedial maintenance responsibilities have been assigned to an in-house computer maintenance group. This group provides cost-effective 20-hour, 5-day on-site coverage and on-call standby coverage for after hours. To maintain the reliability goals, preventive maintenance is very closely controlled, and online diagnostics and analysis are emphasized. A primary component in the reliability and availability of the hardware is the physical environment in the computer facility itself. Accordingly, significant time and resources are expended in ensuring that the best, most cost-effective environmental controls are at the facility.

System Software

The software group's overall goal is to install and maintain, at maximum reliability, ISI's VMS, UNIX, and TOPS-20 operating systems and applications software. In order to accomplish this goal, the group provides 24-hour, 7-day coverage to analyze system crashes and to provide appropriate fixes. In addition, it is the group's responsibility to install, debug, and modify the latest monitor and kernel versions, and the associated subsystems, available from the vendors.

Operations

The operations staff is responsible for operating the computers and watching over the systems and the environment. Operations at the facility runs on a 24-hour, 7-day on-site schedule. One of the primary responsibilities of the group is to provide protection and backup for user files in order to ensure the integrity of all data. This goal is achieved through a variety of means, including regularly scheduled full and incremental backups of all systems; permanent archivals of requested or infrequently accessed system and user files; magnetic tape storage and pointers to all information extant at the time of removal of directories from the various systems; and, perhaps most important, redundant offsite storage of all significant information active on the disk structures or existing on tape within the facility.

When a problem occurs, the on-duty staff isolates it and takes appropriate action. On the night and weekend shifts, the operations staff responds directly to user inquiries. Proper training, experience, continuity and familiarity with the environment are especially stressed.

Network Services

Network Services, the ISI customer-service group, provides a two-way communication link between the users and the rest of the support staff. This is accomplished by maintaining a 12-hour, 5-day on-duty staff for prompt problem resolution and rapid information exchange, both on-line and by telephone. The group offers introductory

training in the use of hardware and software tools available on the ISI systems, as well as providing documentation for new users of the Internet. Network Services also assists in the formulation of user training programs for large, Internet-based military experiments at, for example, the Strategic Air Command, Offutt Air Force Base, Nebraska; the Naval Postgraduate School, Monterey, California; and the Systems Design Center at Gunter Air Force Base, Alabama.

Appropriate documentation is constantly being generated and distributed to individual users, as well as to remote user-group liaison personnel; this documentation ranges from simple, beginner-level explanations to more technical information suitable for experienced users. In accordance with ISTO guidelines, the customer-service group provides regular accounting data to DARPA.

11.4 PROGRESS

The past year has been a successful one for the Computer Center. Average uptimes were 99+ percent, and several major transitions have occurred smoothly and--more important--with virtual transparency to the user community. The most significant transition over the last year was the move of the users of the ADA Joint Program Office from the USC-ECLB machine to ISIF (which was in turn renamed to ADA-20.ISI.EDU). The renaming of all ISI hosts with the new domain-style conventions was also accomplished.

Hardware Additions

Over the past three years, the ISI Computer Center has undergone a major change in direction. Previously dominated by DEC PDP-10 computers, the facility was a comparatively simple support environment consisting of a uniform collection of hardware and software. The current collection of equipment reflects the New Computing Environment project plan of moving toward personal workstations and rear-end servers. The new collection of systems is a substantially more complex support problem involving three local area networks (in addition to the ARPANET/MILNET connections), twenty different main processors (some of which have evolved into powerful, tightly clustered configurations with the acquisition in the last year of an additional DEC 8650 processor), and three major operating systems.

The Computer Center also houses and/or lends substantial support to the array of personal computers (PCs), workstations, and symbolic processing engines acquired over the last several years by a variety of projects. The Center assures that the filesystems of these various machines are backed up and that Network connectivity (where required) is robust, in addition to physically housing major portions of most ISI systems.

The current list of major multiuser processors, network servers, and supported individual machines (PCs, workstations, and symbolic processing computers) follows:

Quantity:	*Manufacturer:*	*Model:*	*Operating System:*
6	DEC	PDP-10	TOPS-20
1	DEC	VAX 8650	VMS
1	DEC	VAX 8650	UNIX
2	DEC	VAX 11/780	UNIX
1	DEC	VAX 11/780	VMS
2	DEC	VAX 11/750	UNIX
8	DEC	VAX 11/750	VMS
4	BBN	C-30 IMP	UNIX
12	XEROX	8010	STAR/MESA
3	XEROX	1100	Interlisp based
3	3-RIVERS	PERQ	PERQ
17	SYMBOLICS	3600/3640/3645/3670	QLISP based
1	LMI	LAMBDA 2x2	QLISP based
12	TI	Explorer	QLISP based
10	HP	Bobcat	UNIX
15	SUN	2/100,2/120,2/150	UNIX
15	SUN	3/50,3/110,3/160	UNIX
40	IBM	PC/XT/AT	MS-DOS

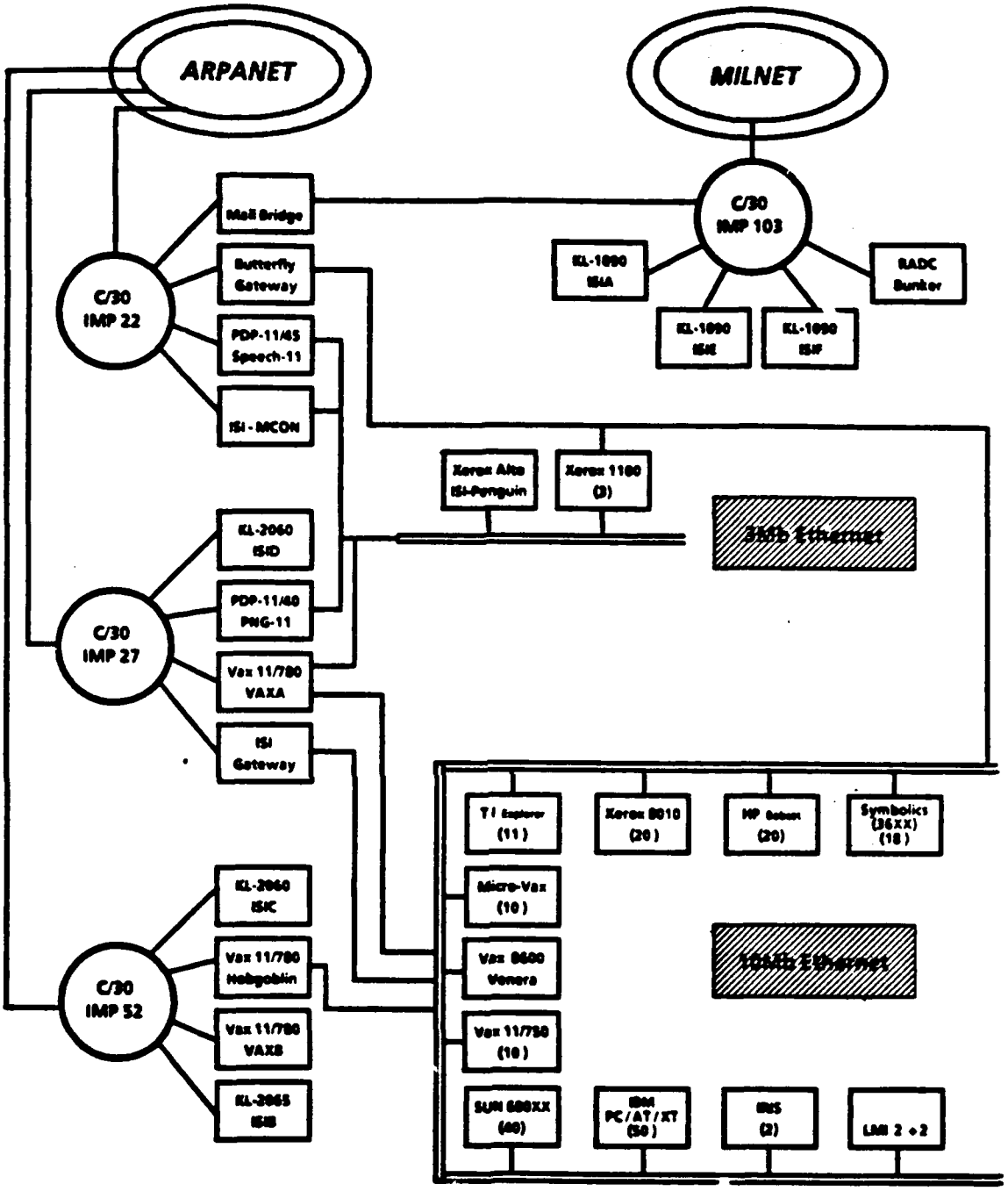
A map of the major pieces of the current local hardware configuration is shown in Figure 1-1.

System Software Enhancements

During the past year, one of the major efforts continued to center upon making the above collection of hardware more compatible from a software point of view. Part of the work involved continued development and debugging of TCP/IP for the various operating systems as well as writing and installing new software for hardware that had either no TCP/IP at all, or no other robust method of communicating with the other systems. One of the most notable of these efforts continued to be on the IBM-PC.

Of equal significance has been the integration of the SUN workstations into the environment. The acquisition of a large array of grant equipment from Texas Instruments and Hewlett-Packard into the environment has also had a substantial impact on the researchers at ISI and on the support staff.

Other significant development work over the last year included accounting systems enhancements on TOPS-20, upgrades to the accounting systems on UNIX and VMS, and numerous upgrades and bug fixes made on many System and user subsystem programs and utilities.



Computer Facility Configuration

Information Sciences Institute

Figure 11-1: Diagram of local ISI internet facility

11.5 MILITARY IMPACT

ISI is perhaps the finest university-based research center promoting the sharing of software resources within the DARPA community; it assumes responsibility for providing support to key DoD community personnel so as to demonstrate the great utility of the ARPANET and MILNET resources. The effective and rapid transfer of information, electronic mail and other data, and computer programs and tools illustrates, on a working daily basis, ways to enhance military efficiency. Specific technology transfer of relevant research to the military is heavily dependent on the facility.

ISI's computer center provides ARPANET/MILNET cycles and support 24 hours a day, 7 days a week to the Strategic Air Command, Naval Postgraduate School, Gunter Air Force Base, Office of Naval Research, and the ADDS Experimental Test Division at Fort Bragg, NC, as well as to the contractors, researchers, and administrators of militarily significant research coordinated out of the DARPA office in Washington, D.C. In addition to supplying machine time and directed Network Services support, this project continues to provide substantial additional support in the following areas:

- General accounting information as needed for workload analysis on the various machines.
- Rapid response to user requests and problem reports.
- Tailored security awareness, and manual and automated tracking.
- Maintenance and upgrading of electronic mail system software, shared online bulletin boards, and other specialized communications and file-transferral programs.
- Maintenance of approximately 2500 user directories (as well as hundreds of support and overhead directories) on the TOPS-20 machines used by the DoD and affiliates.

11.6 FUTURE WORK

The Computer Research Support project will continue to provide computing service to the DARPA research community, provide and support software packages for the ARPA-Internet community, and offer a program of technology transfer and user education through the Network Services group.

Some specific planned activities for the next year include the following:

- The final phasing over of local ISI staff to the NCE (New Computing Environment), resulting in the potential freeing up of the last TOPS-20 resources to the wider military community.
- Installation of new procedures and support for acquisitions of the NCE and other projects at ISI. This will include potential new products (either

purchased at DARPA direction or via grants) from the following vendors: Hewlett-Packard, Texas Instruments, SUN Microsystems, Symbolics, Digital Equipment Corporation, and others.

- Installation of new releases of VMS, UNIX, and ULTRIX on our VAX computers.
- Installations of new releases of UNIX and other operating system software on ISI's other host computers and workstations.
- Installation of portions of TOPS-20 version 6.0 through 6.2, which will be of use to the ISI user communities.

12. DARPA HEADQUARTERS SUPPORT CENTER

Research Staff:
Dan Pederson
Ray Mason
Dan Trentham

12.1 INTRODUCTION

In FY85, a small support effort for ISTO in the form of SUN servers and a small conference room for Strategic Computing meetings provided the genesis for the *Washington Office* and *Remote Maintenance* projects. These projects blossomed into a fully operational computer room (serving both ISTO and MIS), a large conference room, and remote maintenance services for this facility. As the Remote Maintenance and Washington Office projects are very closely related, this report will address both.

12.2 PROBLEM BEING SOLVED

The DARPA-ISTO offices in Washington, D.C., require a small computer facility to support an array of workstations that provide management, project managers, and support staff with CPU cycles to communicate, produce high-quality reports, etc. Herein lies the Washington Office project.

Accomplishing the above brings up the subject of hardware maintenance. Maintenance of such a small facility does not require a full-time maintenance engineer. As a result, other maintenance approaches were investigated. The problem being addressed here, then, is maintaining hardware without a full-time maintenance engineer on site.

12.3 GOALS AND APPROACH

ISI has set out to supply DARPA with a reliable computer facility that can be maintained in a cost-effective manner. To meet this goal, ISI first surveyed DARPA personnel as to what was needed and wanted in terms of computer power. With the data gleaned from this survey, ISI's hardware and software personnel were able to choose from the commercially available workstations. After much testing, SUN Microsystems' workstations were chosen and installed.

With this installed equipment base, the next problem became cost-effective maintenance of the computing equipment. The goal here was a cost-effective, non-manned maintenance scheme. Using the communications channels available (the

ARPANET, telephone lines, etc.), ISI set out to monitor and maintain the DARPA facility.

12.4 SCIENTIFIC PROGRESS

In FY86, ISI installed two major computers at the DARPA facility: a VAX 11/780 to be used as an ISTO planning system and backup file server, and a VAX 11/750 to be used as a file server for the twenty SUN workstations already on-site. These systems both run Berkeley 4.2 UNIX software. The two existing local area networks (LANs), one for MIS use and the other for ISTO, were extended to allow more users. ISI also installed eight additional dial-up lines and modems for use by the MIS group.

Bolt Beranek and Newman, Inc., replaced the LSI-11 gateway with a Butterfly gateway, which provides greater throughput. This gateway provides an interface between the Ethernet and the ARPANET. On the MILNET side ISI installed additional power, and hardware for more conditioned high-speed lines.

ISI now has comprehensive remote monitoring capabilities for the DARPA facility. The Sparton remote device monitoring multiplexor system allows ISI technicians to monitor the DARPA facility's temperature, humidity, electrical power, halon system, VAX 11/750, and VAX 11/780. It provides immediate notification of power outages, changes in temperature or humidity, halon problems, or VAX system crashes. In order to provide 24-hour-a-day observation of the remote site, we have designed and implemented a system to capture video images at the remote site, transfer them in a compressed, digitized form over the ARPANET to ISI, and display the images on a monitor. The system consists of a standard vidicon camera, an IBM-PC at either end, each equipped with digital image processors and Ethernet interfaces, and a standard TV monitor. The software to drive the image processor was developed locally. The network software was built on top of the MIT TCP/IP package for the IBM-PC, using a simple protocol developed expressly to control the packet transmission. The monitoring system has been in service successfully, providing three views of the DARPA facility on a rotating basis.

12.5 MILITARY IMPACT

Remote maintenance would be of value in many small computer sites that do not require a maintenance engineer. Hardware/software maintenance is, however, required at any site. The problem, then, is to provide support to a small site in a cost-effective manner. The most common solution is to purchase maintenance agreements from the vendors involved.

Multi-vendor maintenance contracts can be relatively costly. Response time is another

consideration, especially for sites not near major metropolitan centers. Maintenance personnel must sometimes travel great distances to reach such sites, causing long downtimes for the affected equipment. There is also the problem of "finger pointing" in such an environment, where vendors blame one another for problems of a more difficult nature. Security is also an issue, as each site must be accessible to its maintenance personnel.

Remote maintenance for these sites would allow the handling of many routine hardware/software problems from a remote location. In theory, such an approach would save both time and money by providing highly trained technical personnel in a central location, ready to handle problems as they arose, and doing so remotely whenever possible. Such personnel would always be available in the support center, thus cutting response time to virtually zero. As support personnel would be used by the remote sites on an "as needed" basis, maintenance costs would be held to a minimum. Remote maintenance would assume total responsibility for a given site, eliminating "finger pointing." Security problems would also be held to a minimum, because most problems would be solved remotely, in conjunction with on-site personnel. Some severe problems would require a central support person to travel to the remote site, but these instances would be rare. In such cases, where parts were needed, the primary support site would be able to furnish parts from a large store, thereby significantly reducing downtime. Another advantage of remote site maintenance is that the supporting facility can provide around-the-clock surveillance of the remote site. Such observation reduces the need for after-hours operators.

12.6 FUTURE WORK

Because of the close relationship between the Washington Office and Remote Maintenance projects, they will be combined under the name DARPA Headquarters Support Center. This project will complete the preparation of the DARPA facility. ISI will establish maintenance contracts and schedule preventive maintenance of existing equipment. Additional electrical power will be installed as needed. ISI will make new cables for the long-haul modem cabinet to allow the doors to close properly. A local modem security system will be installed on the VAX dial-up lines to prevent unauthorized access to the ISTO VAXes. Emergency procedures will be finalized and documented for the facility and will be made available to relevant DARPA personnel.

Finally, the problem of water condensation on the computer room windows will be solved. Two plans are under consideration: 1) installation of double-pane windows (cost:~15K). and 2) complete insulation of the computer room (cost:~7.6K). Consultants have recommended the latter solution, but no determination has been made.

Remote maintenance plans include a central communication point for selected equipment. This communication point, or Control Switch, will provide a consolidation of remote communication with the DARPA facility from ISI.

The Control Switch is a DEC uVAX II with eight RS-232 communication lines and an Ethernet interface. The eight communication lines will be connected to the following equipment, with the uVAX providing file capability, gateway communication capability, or both:

- a DEC VAX 11/750 system used as a file server for the SUN workstations. Communication with this system provides remote console capability for file system maintenance and remote hardware diagnostic service of the VAX 11/750.
- a DEC VAX 11/780 backup file server and ISTO planning system. Communication with this system provides the same capabilities as for the VAX 11/750.
- a Sparton remote device monitoring multiplexor. Communication with this system provides for communication of environmental data and VAX system status information to the central monitoring site (ISI). This includes temperature, humidity, power, and halon data, and crash alarm status for the VAXes.
- an NCS card key system, which regulates access to the computer facilities.
- a LeeMah Datacom security system, which can protect ISTO and MIS dial-up lines through the use of a password and dial-back scheme. The uVAX will act as a communications gateway and a database storage source for the card key and LeeMah Datacom security systems.

Information to and from the remote monitoring site will travel through the ARPANET, an ARPANET/ISTO LAN gateway, the ISTO LAN, and to and from the uVAX. A second path between ISI and the uVAX will be a dial-up line that will provide backup in case of failure of either of the networks, or if network loading makes response time unacceptable. See Figure 12-1 for a schematic of systems and communication paths.

The uVAX will also monitor the ISTO LAN and will assist in isolation of problems.

Additional power will be installed for new equipment, maintenance contracts will be negotiated, and additional facility work required by ISTO will be done.

Maintenance of equipment in the facility will consist of the following:

- Power distribution unit (quarterly). This will include checking for grounding problems, verifying that the unit shuts down properly, tightening all hardware to ensure proper connections, and reviewing new needs for additional power cables.
- Security systems (quarterly). This will include checking door locks for proper operation, tightening all electrical connections at the doors and control boxes, verifying that the door security and modem security databases can be down-line loaded, and verifying proper modem operation for remote access to the systems.
- ISTO VAXes (quarterly). This will include changing filters, verifying power-supply adjustments, checking disk drives for amount of soft errors, checking tape drive, checking CPU memory for soft errors, installing FCOs to ensure that systems are at the latest revision levels, and installing diagnostics updates.
- Computer room cleaning (quarterly). This will include thorough cleaning under the floors, complete wipe-down of all equipment, and thorough cleaning and waxing of the floor tiles.
- Halon and fire alarms (quarterly). This will include verifying proper pressure of halon bottles and halon fire extinguishers, and testing the main control panel and the abort button.
- Air conditioning units, both computer room and conference room (monthly). This will include changing filters, cleaning pans, checking freon levels, and checking the condensers on the roof.

13. EXPORTABLE WORKSTATION SYSTEMS

Research Staff:
Dale Chase
Jim Koda

Support Staff:
Manon Levenberg
Ray Mason
Dan Trentham

13.1 PROBLEM BEING SOLVED

The use of dedicated, high-performance workstations and associated file servers offers substantial advantages over terminals connected to time-sharing mainframes. The ever-increasing availability of integrated software environments has greatly facilitated the management and tracking of complex contracts and budgets for DARPA's program managers.

ISI provides DARPA/ISTO with the systems installation, configuration, and management expertise to successfully integrate new workstations and software into the ISTO computing environment.

13.2 GOALS AND APPROACH

The primary goal of this project was to install and support a workstation environment at the DARPA-ISTO offices in Arlington, Virginia. This environment is also intended to provide a testbed for several other concepts and facilities. The environment is configured to accommodate a variety of workstation models with varying capabilities. This configuration provides a minimum set of services on each workstation model:

- document preparation
- mail handling
- file storage and manipulation
- administrative activities (spreadsheets, calendars, etc.)

These services are provided either directly on the workstations, or on the central server (a VAX-11/750). All of the workstations are connected to a local Ethernet and to the ARPA-Internet via a BBN-maintained gateway, and by equivalent code on the VAX that functions as a gateway in the event of failure.

To provide the DARPA-ISTO program managers with direct exposure to delivered software, we work closely with the developers at other DARPA-sponsored centers to handle installation, customization, and problem resolution within the ISTO environment.

13.3 SCIENTIFIC PROGRESS

Over the last few years, ISI has been the primary supporter of the computing environment at DARPA/ISTO. We have installed numerous SUN Microsystems, Inc., workstations, upgraded the VAX file server at ISTO from the 11/750 to the 11/780, maintained and upgraded the SUN Operating System as new versions have been released by SUN, and installed, upgraded, and maintained numerous high-resolution laser printers and printer software. In addition to the standard UNIX environment that SUN provides with their workstations, we have purchased and installed various third-party software systems for ISTO's internal use. These packages include spreadsheet software such as QCalc and NQCalc, integrated document preparation systems such as Scribe and Interleaf, text editors such as Gnu Emacs, and database management systems such as Ingres and Unify.

A significant part of the ISTO support effort has been to customize software packages used at ISTO to meet their operating requirements. ISI has worked to modify the MM mail package from SRI to meet ISTO needs, customized the Interleaf document processing system, modified certain features of public domain software used at ISTO, and customized the laser printer driver software to fit the computing environment. ISI has also served ISTO in an advisory role for evaluation of new software packages and new technologies.

13.4 IMPACT

The early acceptance of the workstations now in place at ISTO and the ease of adoption by previous users of time-sharing systems clearly demonstrates the applicability of this new technology to office and administrative environments. This project has created considerable interest among our other user communities, military contractors in particular, as a means of providing increased computational capacity and enhanced functionality.

It is clear that dedicated personal workstations are superior to remote time-shared resources. The major advantages are greatly increased responsiveness of the system, elimination of delays caused by the Internet, and enhanced functionality available by virtue of the large format display and mouse pointing device offered on the workstation.

The ability to integrate additional workstation models into the environment, allowing them to function in an integrated manner, will greatly enhance the portability and transfer potential of research efforts being carried out by most of the DARPA-sponsored groups around the country. The sponsors will be able to run the developed software on identical hardware without waiting for it to be adapted to another system.

13.5 FUTURE WORK

ISI will continue to maintain, support, and enhance the total computing environment at ISTO with additional software targeted at management requirements, and new hardware and system upgrades as ISTO personnel and operational requirements change. ISI will install new workstations, assess the ability of new workstation software to meet ISTO requirements, investigate alternatives to the current VAX file server (whose function can be assumed by current SUN file server technology), and continue to upgrade and refine the computing environment's network software. Because the computing environments at ISI and ISTO are very similar, ISI will be able to test and experiment with all hardware and software upgrades to the ISTO computer center locally before installation at DARPA. This will ensure minimal downtime at ISTO for future upgrades. Many of the software updates will be installed directly over the ARPANET, allowing the update process to occur without disrupting any of the systems operating requirements at ISTO.

ISI will design, deliver, integrate, and customize software in the ISTO computing environment that will allow document interchange between document processing packages such as Interleaf, spreadsheet software, and database software packages. The design specifications for this document interchange software will be developed with ISTO and will be subject to a pre-delivery design review. Software will also be delivered to ISTO that will allow documents developed using the Interleaf document processing system to be interchanged between SUN workstations, Apple MacIntosh II, and IBM-PC computers.

ISI will maintain all hardware equipment. If a problem occurs, the ISI headquarters operations staff will take immediate action to remotely correct the situation. If the problem requires action local to DARPA headquarters, the ISI on-site technician will be immediately notified and further steps taken to resolve the problem. It is expected that in 95 percent of the cases, corrective action will be achieved within 1 to 3 hours. In those rare cases where problem resolution goes beyond that time because of lack of appropriate spare parts or on-site skills, parts and/or technicians will be dispatched from ISI within 24 hours. This strategy is extremely effective because of the combination of very reliable equipment, remote monitoring, an on-site technician, the large backup of spare parts, and in-depth expertise available upon short notice from ISI. Over the years, ISI has demonstrated that the costs of this strategy are far less than the cost of procuring separate maintenance subcontracts.

ISI will support the DARPA/ISTO staff with a hot-line and electronic mailbox for all inquiries, requests, and problem reports. All messages received from DARPA/ISTO will be answered within 24 hours. If the request will take longer than 24 hours to complete, an estimate of the time involved and completion date will be included in the reply. All

incoming requests and inquiries will be tracked by ISI's Action Accounting System. Every week, or requested period, a report summarizing the status of each open ISTO inquiry/request will be sent to DARPA/ISTO. This reporting mechanism will be coupled with the Action Accounting System to provide DARPA/ISTO program managers with quality tracking information on open ISI action items.

14. NEW COMPUTING ENVIRONMENT

Research Staff:
Dale Chase
Jim Koda

Support Staff:
Glen Gauthier
Manon Levenberg
Ray Mason

14.1 PROBLEM BEING SOLVED

For the past decade, the computational needs of researchers in all areas of computer science have been provided on large-scale timesharing machines. This is no longer a viable approach, and the availability of high-speed, dedicated workstations offers substantial advantages over timesharing. ISI's two DEC mainframes running TOPS-20 are at full capacity and have become very costly to operate. Additionally, they are not capable of supporting many of the current and proposed research efforts at ISI.

As techniques and expectations (especially in the area of Artificial Intelligence) have advanced, general-purpose machines are no longer able to provide the style of interaction and capabilities, in terms of throughput and address space, that are required to support current research efforts. The NCE project continues to:

- Provide a very significant improvement in both the amount and quality of available computing resources to the ongoing ISI research efforts through the use of dedicated personal workstations and higher capacity centralized processors and servers.
- Provide for diverse access methods to the common set of services.
- Fully support the use of special-purpose workstations and processors as required by individual research projects.
- Free up capacity on the existing mainframes by offloading some common functions to central servers.

14.2 GOALS AND APPROACH

The ISI research community is made up of a diverse set of projects and cultures. In order to best support a community with such varied needs, we decided to implement a heterogeneous environment consisting of a variety of personal workstations, dedicated small mainframes, and continuing use of our existing timesharing systems. The New Computing Environment was designed to consist of local processing nodes (workstations and multiuser mainframes) connected via a local network to a set of central servers. A certain minimum set of functions must be available to all nodes in this environment: mail, file manipulation (access and transfer), and Internet connectivity (Telnet). The environment provided by these nodes and servers conceals the fine-grain detail from outside users; users external to the environment need not know what workstation a

particular user is on, unless they want to. At minimum, these services should be available via any of several communication media: local area network, Internet, and dialup access. The NCE provides additional support as needed to allow low-end workstations and terminals on our mainframes to use these services as easily as the fully capable workstations in the SUN, DLion, and Apollo class. Primary access to these services is via the DoD IP/TCP protocol suite, with some extensions and enhancements to support particular modes of interaction that are not yet supported (e.g., random file access and remote procedure call).

The use of these services by our existing mainframes has resulted in immediate improvements for all users, as many CPU-intensive activities have been offloaded to central servers. Making these services available from our mainframes also provides for their use from home terminals. Access for these devices is via dialup lines to our existing mainframes, or to the servers themselves.

The following services are (or will be) provided to all nodes in the environment:

- centralized mail service
- centralized file service
- document formatting
- high-quality printing
- communications
- specialized processing

While the principal incentive for the New Computing Environment was the support and integration of a diverse set of dedicated personal workstations, not everyone at ISI requires this kind of hardware and capabilities. Until they have an actual need to migrate their research efforts to workstations, there will be a significant population content with our existing mainframes (KLs and VAXes). However, the enhanced facilities offered by the provision of centralized servers also benefits this population. These enhancements take the form of increased accessibility of files and mail, increased reliability in terms of both data integrity and security, and improved performance of the existing mainframes due to the offloading of CPU-intensive activities to the dedicated servers.

To reduce the operating and maintenance costs of in-house computing, we have moved groups of users from TOPS-20 to VAXes running both UNIX and VMS. The TOPS-20 capacity freed by these transitions will be made available to other DARPA contractors, as determined by the program managers at ISTO.

14.3 SCIENTIFIC PROGRESS

The main computing resource at ISI, a DEC VAX 8650, was updated to run Berkeley UNIX, version 4.3. We installed a number of SUN-3 workstations and file servers purchased under the second round of NCE acquisitions. Connectivity with the Xerox File Server and Printer was achieved using the XNS support in 4.3 bsd UNIX. A new version of the TOPS-20 operating system was installed to support access to the Imagen laser printers via the Internet.

The initial design and development of a system that makes existing archived files available over the Ethernet to any other system was completed and successfully installed. Work also continued on the integration of the Massbus Ethernet System into TOPS-20. Once this is completed, TOPS-20-based resources will be available on the local area network.

Work also continued on the Laser Disk Archive system that will be used at ISI and ISTO.

14.4 IMPACT

Our experience with personal workstations has made it clear that they represent a useful alternative to large, timeshared mainframes for certain research activities. Several projects within the institute have already moved the majority of their work to workstations, and have therefore been able to continue research that exceeded the capacity of our mainframes. The timely support provided for different workstation models has increased the acceptance of a distributed model of computation.

As the reliability of added facilities and of the entire environment are proven at ISI, we offer the same type of environment to outside contractors via the Exportable Workstation Systems project. These two projects combine to provide an ideal environment for developing and refining the facilities and capabilities that are becoming increasingly important in the command and control context. These capabilities include the handling of redundant databases and the interoperability of a diverse collection of hardware.

14.5 FUTURE WORK

We will identify the computing needs for the remainder of the user community that has yet to make the transition from centralized computer resources to workstations, and then plan configurations and procurements to meet these needs. We will also study subnet configurations of the local area network to optimize network traffic. To aid in this study, we plan to implement an extended network traffic analyzer that will provide detailed short-term logs and long-term traffic histograms.

The Laser Disk Archive project will be completed and installed at ISTO. We plan to study approaches that will allow remote workstations to gain access to network-based resources via dial-up lines and high-speed dedicated links.

We will also specify and implement a full-function mail service for IBM-PC's. The service will be integrated into the local area network and will interface to Internet mail services.

15. STRATEGIC COMPUTING - DEVELOPMENT SYSTEMS

Research Staff:
Ron Ohlander
Stephen Dougherty

15.1 PROBLEM BEING SOLVED

Using recent advances in artificial intelligence, computer science, and microelectronics, DARPA plans to create a new generation of "machine intelligence technology." The DARPA Strategic Computing program will be targeting key areas where advances can be leveraged towards broad advances in machine intelligence technology and the demonstration of applications of the technology to critical problems in defense.

The Strategic Computing program will be supported by a technology infrastructure. This infrastructure will serve to bootstrap the program by providing current state-of-the-art computing technology, network communications, and shared resources to the selected program participants.

The aim of this project is to provide development computers for the Strategic Computing program, to provide system integration as required, to disseminate the systems to the program participants, and to define an architecture for system communications and resource sharing among the computers acquired.

One of the project's aims, which has only been clarified over the last year, has been to provide cost-effective and state-of-the-art engineering workstations to the Strategic Computing community. As the research in this program has progressed, and as the sophistication and complexity of available hardware and software in the vendor community have evolved, the more generalized workstations have proven to be a valuable adjunct to the already established base of specialized symbolic processors.

15.2 GOALS AND APPROACH

A number of machines have been developed to support high-speed processing of large symbolic programs. Each of these machines provide an extensive interactive programming environment, a sophisticated display-manager "window system," a real-time, window-oriented editor, incremental compilers, and dynamic linking.

These systems are the state of the art in program development environments. They are widely used in the research community for systems development, expert systems,

natural language systems, mapping applications, and support of CAD/CAM environments.

Several problems are associated with using these systems as a result of their high individual costs. Since they are single-user systems, they cannot be timeshared in the traditional sense. However, because the machines are too expensive to be supplied on a one-to-one basis to researchers, workers are currently placed in the awkward situation of having to schedule their computing needs. The resultant scheduling conflicts naturally lead to low researcher productivity.

An examination of dynamic machine use shows that many activities do not require the high-speed processing capabilities afforded by these machines. Less expensive, general-purpose machines supporting the same language base have become available only over the last two years. While formerly adequate only for less-intensive research activities such as editing, text preparation, and file scanning, these general-purpose machines are now sufficient for most of the more intensive program development and execution activities.

The parallel duality in resources required for different types of activities and the availability of machines of appropriate cost/performance for carrying out those activities suggests a workstation/server architecture based on these two classes of machines and an interconnecting network. In order to support dynamic source code exchange between server and workstation, it is necessary for each to support the same AI language system. Common LISP is the natural choice in such an architecture, as it was designed with the concepts of commonality and portability as primary goals.

A significant fallout of such an architecture, when viewed as workstations and servers on IP/TCP-based networks, is the ability of researchers to communicate over the Internet to use high-powered resources available on prototype and low-production machines. This has been particularly useful on prototype machines being developed under the machine architectures phase of the program.

ISI has acquired a mix of all of these types of machines to support the requirements of the Strategic Computing program, and has helped to integrate and test individual systems as required for particular Strategic Computing program participants.

The integration tasks have avoided duplication of effort among Strategic Computing developers. ISI has collected software to support this architecture as it is has been developed by vendors and the research community. ISI does a modest amount of testing, modifying, and augmentation of software, provides support for software exchange and distribution among DARPA development sites, helps assure working compatibility of the Common LISP systems, works with commercial vendors to resolve

vendor-specific problems, and works to allow network compatibility of the systems with DoD network protocols (IP/TCP). The software and system integration efforts are carried out at the ISI Marina del Rey facility.

15.3 SCIENTIFIC PROGRESS

During the reporting period, additional symbolic-processing and general-purpose configurations were acquired. Some of them were installed and integrated for several months at ISI, while most were shipped out to a number of research facilities, educational sites, and government agencies.

In the past year, 81 machines were acquired as part of this Strategic Computing project, as well as a large variety of component peripherals, software packages (from Interleaf, CCA, Unipress, etc.), and software maintenance agreements. The major machines acquired were:

- 36 - Symbolics - 3645/3675
- 1 - Texas Instruments - Explorer
- 11 - SUN - various models
- Various hardware peripherals packages

Several of these machines are in use at ISI in preliminary work on natural language and the Common LISP Framework effort.

15.4 IMPACT

The DARPA Strategic Computing program will continue to develop and integrate advanced computer technology into military applications. Technological advancement will be sought in the areas of microelectronics and high-speed symbolic machines, as well as application of this technology to the military. Potential military applications of this technology include autonomous systems (land, air, and sea), battlefield management and assessment, planning and advising, and simulation systems.

The initial program applications included an autonomous land vehicle, a pilot's associate, and a carrier battle group battle management system. These applications test the evolving technology in the areas of vision, expert systems, speech recognition, and high-performance knowledge-processing systems. Each application seeks to demonstrate the new technologies' potential for providing a major increase in defense capability and to reflect a broad demonstration base that is relevant to each of the three services.

The development systems acquired through this project will support the technology base and the targeted applications under the Strategic Computing program.

15.5 FUTURE WORK

Several corporate system integrators and chip manufacturers have perceived that there will be a substantial marketplace for symbolic processing engines in the future. We expect that the price/performance ratio of these new machines will continue to improve at a rapid pace, resulting in more efficient use of quality researcher time as more state-of-the-art machines can be acquired at the most reasonable price.

During the next year, ISI will continue to negotiate with qualified vendors and acquire a mix of high-end LISP machines and other engineering workstations capable of handling symbolic processing and other applications. As the workstation/server architecture is defined, ISI will configure systems, test vendor software, and distribute systems to Strategic Computing program participants as required and directed by DARPA.

16. STRATEGIC C3 SYSTEM EXPERIMENT SUPPORT

Research Staff:
Stephen Dougherty
Victor Ramos

16.1 PROBLEM BEING SOLVED

DARPA has defined an experiment in Strategic C3 systems to be conducted in cooperation with the World Wide Military Command Control System (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC). The concept of the experiment is to demonstrate and evaluate the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge base techniques) for strategic command, control, and communication.

ISI's portion of the plan is to provide an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with this technology and to ensure the success of the experiment. Specifically, SAC must have fairly broad-based experience with ARPANET-based online interactive computing. Installation and maintenance of modems and of 60 or more interactive CRT terminals, user training, system software training and support, and on-site maintenance of equipment for the 1000+ users at SAC Headquarters at Offutt Field, Omaha, Nebraska and at a number of distributed SAC bases, are part of the continuing program.

16.2 GOALS AND APPROACH

The total specific goals of this DARPA experiment are to:

- demonstrate and evaluate the survivability of multinode computer-communication networks, including the ARPANET and packet radio, especially for remote access from both airborne platforms and surface sites.
- explore replication and reconstitution of critical knowledge bases on this network in the face of a loss of a large number of links.
- demonstrate and evaluate the rapid reconstitution of a network by rapid deployment of packet radio nets to reestablish connectivity between surviving elements of the network.
- conduct experiments and exercises to evaluate the utility of such a network on a distributed knowledge base to support post-attack C3 activities.

The experiment is defined as having three phases:

1. Phase I is planned to demonstrate air-to-surface packet radio links and gateways into the ARPANET/MILNET as a first step in evaluating the feasibility of a truly survivable strategic network.
2. Phase II is directed toward creating a survivable message system and data bases through multiple copies of the critical components and data across the ARPANET/MILNET.
3. Phase III will address the feasibility of rapid reconstitution of a strategic network by deployment of packet radio networks to reconnect surviving elements of the network.

16.3 SCIENTIFIC PROGRESS

The SAC user community has been growing since the inception of this project. As a result, ISI has been challenged to provide sufficient resources as the demand has spiked upward. Additional Terminal Access Controllers (TAC's) were installed to support the growing user community; additional lines and modems were also added to the Datacommunications base at Offutt Air Force Base. The complete dedication of the Digital Equipment Corporation model KL-2060 host computer, named E.ISI.EDU, to the SAC community helped eliminate user frustration and increase productivity. This was due to the availability of the additional computational capacity. ISI installed and now maintains terminals and communication equipment for the access of various Air Force bases to the MILNET, allowing more widespread participation in the experiment as well.

During this period new resources were also obtained or redistributed to support the VAN van effort (the effort is a portion of the experiment testing the mobility and reconstitutability of a partially destroyed network). Additional ARPANET/MILNET access was provided to SRI International (which was working the Telecommunications portion of this effort and housed the vans) via 9600 bps communication lines.

16.4 IMPACT

One of the premises of this experiment is that SAC personnel will become more proficient and knowledgeable users of the current computer technology, available within the ARPANET/MILNET arena. This knowledge will allow SAC to make more effective evaluations of new technologies for strategic use, to identify areas of potential future research, and to implement those technologies found appropriate.

16.5 FUTURE WORK

ISI will continue to assist DARPA in planning this program, working to satisfy the communication and computer resource requirements of SAC Headquarters. In particular, ISI will do the following:

- Continue to provide on-site maintenance support for the required equipment.
- Continue to plan and assist in implementing improved SAC connectivity to the MILNET/DDN/ARPANET.
- Maintain terminals and communication equipment for the connection of several Air Force bases to the MILNET, allowing increased participation in the experiment.
- Continue to supply computational, programming, and administrative support to users at SAC Headquarters via the resources of the E.ISI.EDU computer, the Systems Programming, Operations, and Network Services staff in Marina del Rey, and the on-site technician.

ISI's most visible direct support will continue to be the on-site technician at SAC, who will be responsible for the identification of system malfunctions and for primary maintenance of on-site equipment. He will be supplied with required spare parts and will have maintenance contracts with the equipment vendors. Further support will be available from ISI in terms of additional spare parts, systems expertise, and documentation as necessary. The on-site maintenance technician will also be responsible for the off-site terminals at Vandenberg Air Force Base, Barksdale Air Force Base, March Air Force Base, and other locations as dictated by the requirements of the experiment. The on-site technician will coordinate data communications requests of SAC AD with SRI International and SAC under the supervision of ISI management. The technician will provide training and consulting to SAC personnel with backup from ISI as required.

ISI will provide program planning assistance to DARPA. We will continue to investigate the data communications requirements for SAC Headquarters, the HERT effort, and the ACCESS system (SAC Automated Command and Control Executive Support System).

PUBLICATIONS

1. Balzer, R., "Automated enhancement of knowledge representations," in *IJCAI 85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 203-207, IJCAI, Los Angeles, August 1985.

2. Balzer, R., T. E. Cheatham, Jr., and C. Green, "Software technology in the 1990's: Using a new paradigm," in W. W. Agresti (ed.), *New Paradigms for Software Development*, pp. 16-22, IEEE Computer Society Press, 1986. Also published in *IEEE Computer*, November 1983, pp. 39-45.

3. Balzer, R., N. M. Goldman, and D. S. Wile, "Operational specification as the basis for rapid prototyping," in W. W. Agresti (ed.), *New Paradigms for Software Development*, pp. 116-132, IEEE Computer Society Press, 1986. Also published in *ACM SIGSOFT Software Engineering Notes*, December 1982, pp. 3-16.

4. Balzer, R., "Transformational implementation: An example," in W. W. Agresti (ed.), *New Paradigms for Software Development*, pp. 227-238, IEEE Computer Society Press, 1986. Also published in *IEEE Transactions on Software Engineering*, January 1981, pp. 3-14.

5. Balzer, R., "Living in the next generation operating system," in *Proceedings of the 10th World Computer Congress '86*, IFIP, Dublin, Ireland, September 1986.

6. Balzer, R., and N. M. Goldman, "Principles of good software specification and their implications for specification languages," in N. Gehani and A. McGettrick (eds.), *Software Specification Techniques*, pp. 25-39, Addison-Wesley, 1986.

7. Bisbey, R., II, R. Parker, and E. R. Cole, "An inexpensive megabit packet radio system," in *Proceedings of COMPCON Spring '86*, IEEE, San Francisco, March 1986. Also published as USC/Information Sciences Institute, RS-86-166, October 1986.

8. Booth, D., *Multiple Strongly Typed Evaluation Phases*, USC/Information Sciences Institute, RS-86-165, October 1986.

9. Cohen, Danny, and K. Fry, "PCB artwork preparation using E-beam pattern generators," *VLSI Design* 6, (8), August 1985, 82-88. Also published in *Proceedings of the IPC 28th Annual Meeting*, New Orleans, April 1985.

10. Cohen, Danny, "Performance analysis for several voice-entry protocols," in *Proceedings of the Voice I/O Systems Applications Conference '85*, pp. 444-459, AVIOS, San Francisco, September 1985.

11. Cohen, Donald, "Automatic compilation of logical specifications into efficient programs," in *AAAI-86: Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 21-25, American Association of Artificial Intelligence, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-86-175, November 1986.

12. Cumming, S., and R. Albano, *A Guide to Lexical Acquisition in the JANUS System*, USC/Information Sciences Institute, RR-85-162, February 1986.

13. Cumming, S., *Design of a Master Lexicon*, USC/Information Sciences Institute, RR-85-163, February 1986.

14. Cumming, S., *The Lexicon in Text Generation*, USC/Information Sciences Institute, RR-86-168, October 1986.

15. DeSchon, A., *INTERMAIL, an Experimental Mail Forwarding System*, USC/Information Sciences Institute, RR-85-158, September 1985.

16. Feather, M. S., "The evolution of composite system specifications," in *Proceedings, Fourth International Workshop on Software Specification and Design*, pp. 52-57. IEEE Computer Society Press, Monterey, California, April 1986.

17. Feather, M. S., "An incremental approach to constructing, explaining, and maintaining specifications," in *Iteration in the Software Process: Proceedings of the 3rd International Software Process Workshop*, pp. 137-140, IEEE Computer Society Press, Breckenridge, Colorado, November 1986.

18. Feather, M. S., "Session summary: Next steps," in *Iteration in the Software Process: Proceedings of the 3rd International Software Process Workshop*, pp. 129-132, IEEE Computer Society Press, Breckenridge, Colorado, November 1986.

19. Feather, M. S., "Program specification applied to a text formatter," in N. Gehani and A. McGettrick (eds.), *Software Specification Techniques*, pp. 289-301, Addison-Wesley, 1986.

20. Friedman, L., "Controlling production firing: The FCL language," in *IJCAI 85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 359-366, IJCAI, Los Angeles, August 1985.

21. Johnson, W. L., and E. Soloway, "PROUST: Knowledge-based program understanding," in C. Rich and R. Waters (eds.), *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann, 1985. Reprinted from *IEEE Transactions on Software Engineering*, SE-11, (3), March 1985.

22. Johnson, W. L., "Specification via scenarios and fragments," in *Third International Software Process Workshop*, ACM SIGSOFT and IEEE-TCSE, November 1986.

23. Johnson, W. L., "Modeling programmers' intentions," in J. Self (ed.), *Intelligent Computer-Aided Instruction*, Chapman and Hall Press, 1986.

24. Johnson, W. L., *Intention-Based Diagnosis of Novice Programming Errors*, Morgan Kaufmann, Los Altos, California, 1986.

25. Johnson, W. L., and E. Soloway, "PROUST: An automatic debugger for Pascal programs," in G. Kearsley (ed.), *Artificial Intelligence and Instruction: Applications and Methods*, Addison Wesley, 1986.

26. Kaczmarek, T. S., R. Bates, and G. Robins, "Recent developments in NIKL," in *AAAI-86: Proceedings of the 5th National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-86-167, November 1986.

27. Kasper, R., "Systemic grammar and Functional Unification Grammar," in *Proceedings of the 12th International Systemic Workshop*, Ann Arbor, Michigan, August 1985. Also published in USC/Information Sciences Institute, RS-87-179, May 1987.

28. Leiner, B., R. Cole, J. Postel, and D. Mills, "The DARPA Internet protocol suite," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 28-36, IEEE, Washington, D. C., March 1985. Also in *IEEE Communications*, 23, (3), March 1985, 29-34. Also published as USC/Information Sciences Institute, RS-85-153, April 1985.

29. Lewicki, G., "Prototyping and small-volume parts through MOSIS," in *GOMAC-85 Digest*, Proceedings of the Government Microcircuit Applications Conference, Orlando, Florida, November 1985. Also published as USC/Information Sciences Institute, RS-85-160, November 1985.

30. London, P., and M. Feather, "Implementing specification freedoms," in C. Rich and R. Waters (eds.), *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann, 1986. Also published in *Science of Computer Programming 2*, 1982, 91-131, North-Holland.

31. Mann, W. C., and S. A. Thompson, "Assertions from discourse structure," in *Proceedings of the Eleventh Annual Meeting of the Berkeley Linguistics Society*, pp. 245-258, BLS, Berkeley, February 1985. Also published as USC/Information Sciences Institute, RS-85-155, April 1985.

32. Mann, W. C., "The anatomy of a systemic choice," *Discourse Processes* 8, (1), January-March 1985, 53-74.

33. Mann, W. C., and S. A. Thompson, "Relational propositions in discourse," *Discourse Processes* 9, (1), January-March 1986, 57-90.

34. Mann, W. C., and S. A. Thompson, "Rhetorical Structure Theory: Description and construction of text structures," in *Proceedings of the Third International Workshop on Text Generation*, Nijmegen, The Netherlands, August 1986. Also published as USC/Information Sciences Institute, RS-86-174, October 1986.

35. Matthiessen, C., "Representational issues in Systemic Functional Grammar," in *Proceedings of the 12th International Systemic Workshop*, Ann Arbor, Michigan. August 1985. Also published in USC/Information Sciences Institute, RS-87-179. May 1987.

36. Mockapetris, P., and J. Postel, "A perspective on name system design," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 349-355, IEEE, Washington, D. C., March 1985. Also published as USC/Information Sciences Institute, RS-85-152, April 1985.

37. Moses, L., *A Basic Guide to UNIX*, USC/Information Sciences Institute, TM-85-157, October 1985.

38. MOSIS Project, "Chips and boards through MOSIS," in *Proceedings of COMPCON Spring '85*, pp. 184-186, IEEE, San Francisco, February 1985. Also published as USC/Information Sciences Institute, RS-85-149, February 1985.

39. Mostow, J., and Donald Cohen, "Automating program speedup by deciding what to cache," in *IJCAI 85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 165-172, IJCAI, Los Angeles, August 1985.

40. Mostow, J., and W. Swartout, "Towards explicit integration of knowledge in expert systems: An analysis of MYCIN's therapy selection algorithm," in *Proceedings. AAAI-86: Fifth National Conference on Artificial Intelligence*, pp. 928-935. American Association of Artificial Intelligence, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-86-176, November 1986.

41. Neches, R., W. Swartout, and J. Moore, "Explainable (and maintainable) expert systems," in *IJCAI 85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 382-389, IJCAI, Los Angeles, August 1985.

42. Neches, R., W. Swartout, and J. Moore, "Enhanced maintenance and explanation of expert systems through explicit models of their development," *Transactions on Software Engineering*, November 1985. Revised version of an article in *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*. December 1984.

43. Postel, J., "Internetwork applications using the DARPA protocol suite," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 37-42, IEEE, Washington, D. C., March 1985. Also published as USC/Information Sciences Institute, RS-85-151, April 1985.

44. Postel, J., G. Finn, A. Katz, and J. Reynolds, *The ISI Experimental Multimedia Mail System*, USC/Information Sciences Institute, RR-86-173, September 1986.

45. Reynolds, J., J. Postel, A. Katz, G. Finn, and A. DeSchon, "The DARPA experimental multimedia mail system," *IEEE Computer* 18, (10), October 1985. 82-89. Also published as USC/Information Sciences Institute, RS-85-164, December 1985.

46. Smoliar, S. W., "A view of goal-oriented programming," in *Proceedings of COMPCON Spring '86*, pp. 112-117, IEEE Computer Society Press, 1986.

47. Smoliar, S. W., "Operational requirements accommodation in distributed system design," in W. W. Agresti (ed.), *New Paradigms for Software Development*, pp. 133-139, IEEE Computer Society Press, 1986. Also published in *IEEE Transactions on Software Engineering*, November 1981, pp. 531-537.

48. Sondheimer, N. K., and B. Nebel, "A logical-form and knowledge-base design for natural language generation," in *AAAI-86: Proceedings of the 5th National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, Philadelphia, August 1986. Also published as USC/Information Sciences Institute, RS-86-169, November 1986.

49. N. K. Sondheimer, ed., *Proceedings of the Strategic Computing Natural Language Workshop*, USC/Information Sciences Institute, SR-86-172, May 1986. Proceedings of a workshop held at Marina del Rey, California, May 1-2, 1986.

50. Swartout, W., "Explaining and justifying expert consulting programs," in J. Reggia and S. Tuhrim (eds.), *Computer-Assisted Medical Decision Making, Volume 2*, Springer-Verlag, 1985. Reprinted from *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981.

51. Swartout, W., "Knowledge needed for expert system explanation," *Future Computing Systems*, 1986. Revised version of a paper published in *AFIPS Conference Proceedings*, National Computer Conference, pp. 93-98, 1985.

52. Swartout, W., and R. Neches, "The shifting terminological space: An impediment to evolvability," in *Proceedings, AAAI-86: Fifth National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, Philadelphia, August 1986.

53. Swartout, W., "Explanation by design," in *Proceedings of the Second Annual Conference on Artificial Intelligence and Advanced Computer Technology*, 1986.

54. Swartout, W., "Beyond XPLAIN: Toward more explainable expert systems," in *Proceedings of the Congress of the American Association of Medical Systems and Informatics*, 1986.

55. Swartout, W., and R. Balzer, "On the inevitable intertwining of specification and implementation," in W. W. Agresti (ed.), *New Paradigms for Software Development*, IEEE Computer Society Press, 1986. Reprinted from *Communications of the ACM*, July 1982. Also published in N. Gehani and A. McGettrick (eds.), *Software Specification Techniques*, pp. 41-45, Addison-Wesley, 1986.

56. Wile, D. S., "Organizing programming knowledge into syntax-directed experts," in *International Workshop on Advanced Programming Environments*, Trondheim, Norway, June 1986.

57. Wile, D. S., "Program developments: Formal explanations of implementations," in W. W. Agresti (ed.), *New Paradigms for Software Development*, pp. 239-248. IEEE Computer Society Press, 1986. Also published in *Communications of the ACM*, November 1983. pp. 902-911.